Intelligent Tactical Vest

Atif M. Khandaker
Worcester Polytechnic Institute

Jeffrey Michael Ferrandino
Worcester Polytechnic Institute

Jeremy Zorrilla De Los Santos
Worcester Polytechnic Institute

Victor Brisan
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/mqp-all

Repository Citation

This Unrestricted is brought to you for free and open access by the Major Qualifying Projects at Digital WPI. It has been accepted for inclusion in Major Qualifying Projects (All Years) by an authorized administrator of Digital WPI. For more information, please contact digitalwpi@wpi.edu.
Worcester Polytechnic Institute

Victor Brisan
Jeffrey Ferrandino
Atif Khandaker
Jeremy Zorrilla

[INTELLIGENT TACTICAL VEST]

The Intelligent Tactical Vest project is an attempt at solving the crucial issue of police officers becoming injured in the line of duty without the ability to call for help. The system is designed to collect data relevant to the health status of officers using a device built into ballistic vests and process that data into a usable form in order to transmit it to relevant responders.]
## Contents

I. Introduction ........................................................................................................... 5

II. Project Goals ....................................................................................................... 6
   A. Unfeasible Goals ............................................................................................... 6
   B. Short Term Goals .............................................................................................. 6
   C. Long Term Goals .............................................................................................. 7

III. Background Research ....................................................................................... 8
   A. Prior Art ........................................................................................................... 8
      1. Impact helmet – Northeastern University ......................................................... 9
      3. Real-Time impact detection and Object Discrimination for Pedestrian Protection ........................................................................................................... 10
      5. Ballistic impact sensing and display system - US patent # 4305142: .......... 11
   B. Medical Background Research ........................................................................ 11
      1. Polar RS100 ................................................................................................... 12
      2. Timex Ironman Race Trainer Heart Rate Monitor ............................................ 12
      3. Pulse Oximeter LM-800 ................................................................................ 13
      4. PDA cortex Life Shirt .................................................................................... 13
      5. Piezoelectric Film Sensor .............................................................................. 13
      6. Cuff-less PPG-Based Blood Pressure Monitor ............................................... 14
      7. Interview with Professor Mendelson ............................................................... 14
   C. Officer Assault Statistics .................................................................................. 15
      1. Officers Assaulted ........................................................................................... 16
      2. Officers Killed Accidentally ........................................................................... 16
      3. Officers Feloniously Killed ............................................................................ 16
   D. System Concept ................................................................................................. 18
   E. Systems Research ............................................................................................... 18
      1. Load/Force Cells ............................................................................................ 19
      2. Single/dual Axis Accelerometers .................................................................... 20
      3. Piezoelectric Sensors Viability .................................................................... 22
      4. Sensor choices .............................................................................................. 23
      4. Gain Stage Expected Specs .......................................................................... 23

Worcester Polytechnic Institute | I. Introduction
IV. System Components .................................................................................................................. 29
   A. Plate Test Stand ...................................................................................................................... 29
   B. Piezoelectric Testing ........................................................................................................... 33
   C. Analog Application Schematic ............................................................................................ 35
   D. ADC Schematic .................................................................................................................. 37
   E. Logic System Level Implementation .................................................................................... 39
       1. Simulation and Testing ..................................................................................................... 41
   F. Asynchronous Memory Interface Controller .......................................................................... 41
       1. The Control Logic ............................................................................................................. 42
       2. The State-machine ........................................................................................................... 43
       3. Simulation and On-board Functionality Testing .............................................................. 46
       4. PicoBlaze Embedded Core ............................................................................................ 50
   G. Data Input Controller ............................................................................................................ 59
       1. Clock Multiplier ................................................................................................................ 61
       2. Clock Divider .................................................................................................................. 61
       3. Address Counter ............................................................................................................. 61
       4. FIFO Buffer .................................................................................................................... 62
       5. Control Multiplexer ......................................................................................................... 63
       6. State Controller ................................................................................................................ 64
       7. Trigger Controller ............................................................................................................ 65
       8. Simulation and Testing ..................................................................................................... 67
   H. Printed Circuit Board Design ................................................................................................. 68
       1. Definitions ....................................................................................................................... 68
2. Custom Parts ................................................................................................................................. 69
3. PCB Layers ................................................................................................................................... 70
I. Localization Calculations ............................................................................................................. 76
V. Results ......................................................................................................................................... 80
VI. Sources ....................................................................................................................................... 82
APPENDIX I ..................................................................................................................................... 84
APPENDIX II ................................................................................................................................... 102
I. Introduction

The Intelligent Tactical Vest MQP is a project that deals with the design of a system for improving mortality rates among wearers of ballistic vests. The project achieves this purpose by using a sensor grid attached to a ballistic vest or its plates to gather data at each impact, data which will then be stored and then used in trilateration calculations to determine the position on the vest of any impacts, be they from gun shots, stab attempts, or blunt force trauma. Ultimately, this information is useful in improving mortality rates because the project could be expanded to include a way to transmit it to appropriate responders such as emergency medical services if the vest is worn by police officers or field medics if it is worn by soldiers. Other potential system components that could further the project’s goal but that were not implemented are using sensor data to determine impact type, adding sensors for detecting vitals, and tracking the user with GPS.

The project initially tested various types of sensors such as resistive plates, microphones, and piezoelectric strips to determine which sensor would best fit the design's goals. For this purpose, simple test circuits were developed which provided an analog to digital converter with appropriate voltage versus time signals. At the same time, an FPGA was set up to efficiently store the data in a manner usable by the team for analysis.
II. Project Goals

This project's goals were divided into goals the team could finish during the year assigned to the project, and goals that would make the project better but were not feasibly achievable in the limited time period available. Also, at the start of the project, the achievable goals were split up into immediate goals achievable in the early project stages and long term goals achievable during the project's entire length.

A. Unfeasible Goals

As mentioned in the introduction, the purpose of the vest system is to store information so that the information can be sent to some sort of responder, such as paramedics for police officers and field medics for military use. This transmitting of information goal is complex enough on its own that it would take up all of the project’s time and was therefore excluded as unfeasible. Another goal taken out due to its complexity is the GPS localization of the vest wearer.

B. Short Term Goals

Early in the project, the team was focusing on testing the feasibility of various sensors and on setting up an FPGA to collect data for later analysis. For these purposes, the team decided to initially split itself up in order to cover four different aspects of the project: testing apparatus design, analog circuitry design, memory interface design, and core design.
During this stage, various types of accelerometers, piezoelectric strips, and microphones were individually tested in order to find out what sort of gain is needed for each sensor. Since the piezoelectric devices, accelerometers, and microphones provide a similar voltage signal, a single design can be used to test all of these sensors. However, the magnitudes of the signals are vastly different since microphones are expected to output very low voltages, piezoelectrics output very high voltages, and accelerometers somewhere in between. All sensor signals needed to be amplified or decreased into a range from zero to as close as possible to the ADC’s maximum input to get as clear a picture as possible. The simplest design that can achieve this is connecting each sensor to an ADC input through a gain stage that can be changed depending on which type of sensor is being tested at the time.

At the same time, analog-to-digital converters were researched and requested. Then initial VHDL programs to poll the ADC and store the output of the ADC were written. Finally, the design needed a way to communicate with the Nexys2 board through RS232 and pull the information stored from the ADC.

C. Long Term Goals

After the short term goals above were accomplished, the team moved on to working on the yearlong goals. Using the data collected from the system designed in B term, the location and the type of the injury could be calculated and stored as data potentially transmittable to responders. Sections of the overall system could also be implemented to detect whether or not the detected blow penetrated the vest and to detect relevant vital signs.
III. Background Research

Background Research was a critical process which was necessary before building any kind of device. Background research for the intelligent vest involved searching for prior art, officer assaults, medical background, and potential system implementations.

A. Prior Art

Prior art is an important tool in the development process. It saves a lot of time by allowing us to see areas of investigation and research done by others. Prior art research for the intelligent vest involved finding products created for the specific purpose of injury localization and vital sign monitoring.

The previous art on impact detection is important because they contain information on how different concepts were setup, an abundance of data, and reasons why inventors chose different features of their concepts. The ballistic impact detection system offered similar ideas to ours using piezoelectric sensors and acoustic sensors. The piezoelectric sensors are placed on the front of the body and are used to detect force applied to the body, and the acoustic sensors are placed on the front of the body and are able to detect whether the impact occurred on the right, left, or center of the body. The Ballistic impact sensing and display system is an excellent concept of impact localization if we implement piezoelectric sensors in a grid array.
1. Impact helmet – Northeastern University

There is another project in progress at northeastern university which has a similar function to this project. It is a helmet which is capable of impact detection and relaying information in the form of a message on an LCD. The purpose of the helmet is to detect severity of injuries on the head from snowboarding/skiing incidents to alert first responders of the significance of the damage.

Unfortunately it is not revealed how the helmet functions in order to perform the impact detection and any assumptions would be baseless. However, the impact helmet is basically a helmet version of what the intelligent vest is out to accomplish, that is to detect, localize, and report the severity of the injury on the body. From the success of the Impact Helmet project we can derive that it is possible to localize and report injury, and it will be possible to detect and localize bullet impact.


The Shok SpotR is an interesting product which is calibrated to detect if a helmet has been damaged from impact. It is essentially an accelerometer which detects sudden changes in acceleration. If the change is significant enough it sets off an alert to make the user aware of possible damage to the helmet, or even the head.

The product is marketed as being useful because of the inability to necessarily see observable damage to a helmet after an impact. This alerts users that the impact was significant enough, and there is potential significant damage to the helmet. This shows that accelerometers could be used to determine various kinds of impacts on the person wearing the vest. For example, one could
record the sudden change in acceleration after an automobile impact, and post acceleration to determine whether the impact was fatal.

3. Real-Time impact detection and Object Discrimination for Pedestrian Protection

This is currently a project which is in progress by Andrew C. Kim sponsored by the BMW automotive group. The project uses several sensors in the bumper of the car to determine impact and take impulse measurements. Utilizing the different frequency responses of various objects hitting the bumper/sensor and the pressures measure an algorithm is able to compare the data from the impact to previously stored calibration data to determine what type of an object it is.

Real-time impact detection is the primary purpose of the intelligent vest, and it is what is necessary determine the types of impact on the vest. This will however require much calibration in order to have the right base data to compare impact data with. However, this does seem like a simple method for determining different types of impact. Implementing this idea would require much calibration to different types of bullets, knives, and other impacts, and then storing the impulse data in the microcontroller, and upon impact checking the impulse response of the impact to that of those stored to determine exact impact type.
4. Ballistic Impact Detection System - US patent # 7660692:

Detects vibration with piezoelectric film sensing elements and measures the frequency and amplitude characteristics to determine injury. Two sound sensors are also placed on the body to “register high-energy acoustic signatures produced by ballistic impacts.”

5. Ballistic impact sensing and display system - US patent # 4305142:

This device detects the location of a ballistic impact using sensors arranged in a grid. The sensors then pick up on the shock wave generated by the ballistic impact and then the location can be calculated depending on the time each of the sensors is activated.

B. Medical Background Research

In order to determine the future medical capabilities of the tactical vest, research was conducted on prior art. This research will provide with important features that should be incorporated in the product and market value for this product. It would be beneficial to determine which features apply to this product. According to EMT guidelines, the most important things to check on a victim that has suffered a bullet wound are the air ways and respiration of the victim, and checking blood circulation. An examination of prior art shows that most of the portable equipment used to measure vitality measures, heart rate, and O2 count, while a minority of these devices have plethysmograph capabilities.
1. Polar RS100

The Polar RS100 is a device used for improving physical fitness by monitoring various vital signs. Most of these health monitoring tools have a general subset of features, some have features that make them stand out from the rest of the market. Some of the more standard features found in training monitors include heart rate and HRmax. One of the unique features this monitor has, however, is the automatic age-based target zone (bpm / %) which gives an expected heart rate of an individual. The measurement of maximum heart rate can be used as a warning sign for the observing paramedics and the user.

2. Timex Ironman Race Trainer Heart Rate Monitor

This heart rate trainer is very similar to the Polar RS100 but has very unique features which include the capabilities of measuring the recovery heart rate your heart rate after a strenuous activity, during or after your workout. It is also capable of protecting your data from cross talk with other heart rate monitors and electronic interference from exercise equipment. It also has a water-resistant hatch in heart rate sensor accommodates quick and easy battery changes. The most important feature to potentially include in the project would be the cross talk protection, as it would reduce the error rate of the actual measurements. Also it would be wise to create a product that is easily maintainable.
3. Pulse Oximeter LM-800

There are two main types of oximeters: light sensor oximeters and plate oximeters. The Pulse Oximeter LM-800 is of the first type. This type of oximeter is usually placed on the finger and measures SpO2 value and pulse rate. This particular oximeter also has an auto shut-off after idle for 8 seconds and a low voltage indicator.

4. PDA cortex Life Shirt

The PDA cortex Life Shirt is a similar type of product that the medical aspect of this project is attempting to replicate. Its main purpose is far different from the original vision for this project. The Life Shirt is designed to reduce doctor visits of patients under medical trials. The product is capable of measuring pulse, blood oxygen, blood pressure, and temperature. The only thing differentiating the life shirt from other forms of fitness equipment would be is its capability to measure temperature. It would be wise to incorporate this feature into the vest, because of the correlation between loss of blood pressure and temperature.

5. Piezoelectric Film Sensor

An alternative to PPG sensors or oximeters would be to use piezoelectric film sensors to capture arterial wave pulses. The experiment is focused in three directions, the reduction of noise in arterial pulse signals, different configurations, and sensitivity of the sensor. The experiment results concluded that the pressure of the sensor on the skin does reduce the amount of noise measured by the device. This is an important idea to keep in place because if the sensors are
embedded on the vest they won’t be a vast amount of pressure between the skin and the sensor. Another important premise to take from the experiment would be that of the placement of the sensors is also proportional to the amount noise the system measures.

6. Cuff-less PPG-Based Blood Pressure Monitor

Another alternative to piezoelectric film would be to use photoplethysmograph sensors capable of measuring arterial pressure, which can be used to measure the uses blood pressure. The referenced article goes into great detail on the methodologies used to measure blood pressure in a noninvasive manner. The article mentions the use of Oscillometry, and height sensors utilizing accelerometers as well as the mathematics involved in the calculation of the resulting blood pressure.

7. Interview with Professor Mendelson

Yitzhak Mendelson is an associate professor at WPI and has a PhD in biomedical engineering from Case Western Reserve University. Professor Mendelson is one of the leading researchers in Biosensors, microcomputer-based biomedical instrumentation and noninvasive blood gas monitoring. His particular background provides valuable information due to his expertise in the biomedical sensor field and is the reason an interview with him was conducted.

During the meeting with Professor Mendelson, various technologies were discussed that may be used to measure important vital signs. He also discussed which vital signs are important when one is victim of a gunshot wound. According to the professor, the important vital signs to
measure are heart rate, blood oxygen count, and blood pressure. However he did state that creating a non-invasive method for measuring one’s blood pressure is difficult enough to create in a Biomedical Engineering Major Qualifying Project. We also discussed various types of oximeters and locations on the body which would be appropriate to measure O2 in areas where the vests would cover. Professor Mendelson suggested the use of reflective oximeters which are used on foreheads and experiment to see if we can get similar results in the sternum and the cervical vertebrae. Lastly the issue of measuring the user’s heart rate was discussed, he suggested we modify some of the health monitoring equipment and apply it in our project. Another important tip the professor gave was to measure electro cardiogram instead of heart rate as the EKG gives more information to the paramedics.

C. Officer Assault Statistics

In order to see how the vest system could impact the target market, statistics of officers assaulted in the line of duty were looked into, specifically instances where a faster response time would have helped the officer survive. The major source for this research was the Federal Bureau of Investigation's database. The Federal Bureau of Investigation has a large database of statistics with regards to officer assaults that may have resulted in injury or death. The statistics are divided into the categories of officers that were killed feloniously, officers killed accidentally, and officers assaulted. The most recent data is drawn from 2008, with 518 120 officers serving 226 million people, almost 75% of the population, making for an excellent sample size nationally. The database also has data from the year 1999 onward.
1. Officers Assailed

Out of all assaulted officers in 2008, 26.1% were injured. The FBI separates the method of assault in the categories of attacks with personal weapons (hands, feet, etc.), knives or cutting instruments, firearms, and other dangerous weapons. The highest injury rate is from personal weapons at 27.8%, followed by 22.6% from other dangerous weapons, 13.4% from knives or cutting instruments, and 8.4% from firearms.

2. Officers Killed Accidentally

The number of officer deaths in much smaller than the raw number of assaults and are presented by the FBI directly instead of as percentages. 68 officers died accidentally in 2008:

"...39 died as a result of automobile accidents. Thirteen officers were struck by vehicles, 6 officers died in motorcycle accidents, 2 died in aircraft accidents, and 5 officers died during other types of accidents."

3. Officers Feloniously Killed

Of 41 officers killed feloniously in 2008, 35 were killed by firearms, 2 by bombs, 4 by a vehicle, and none by personal weapons, blunt objects or knives and cutting objects. Thirty-six of these officers were wearing their vests.
Of the 5 officers that had their weapon stolen, only 1 was killed with it. However, the FBI reports that 3 officers were killed with their own weapon despite it not being stolen from the officer.

Out of the 36 cases where officers were feloniously killed while wearing their ballistic vests, very few deaths were caused by the failure of the vest. The majority of cases consists of the officer receiving wounds to the head or neck. Some cases that might be considered vest failure are:

- CA officer shot in the chest by a .223 caliber rifle
- LA officer shot in between the side panels of the vest by a .40 caliber hand gun
- PA officer shot in the lower abdomen by a 7.62x39mm rifle
- PA officer shot in the upper torso repeatedly by a .45 caliber hand gun

The database, however, contains no specific data on why the vest allowed bullet penetration in the above cases. In most of the cases mentioned above, officers were shot somewhere outside the area covered by the ballistic vest they wore or were killed by something other than firearms, making the vital signs detection of our design quite important in detecting injury to the wearer. Impact type detection and penetration are still important for the vest failure cases.

Many times the officers that were shot were accompanied by others that called for emergency help, making our system in these cases redundant. Our system is still important in cases where the officer was alone such as:

- the CA officer shot in the chest by a .223 rifle while he was in pursuit of a suspect on foot alone
• a lone IL officer shot with his own gun while in a struggle with a suspect
• a LA officer responding to a burglary alone struck by the suspect's truck
• a LA officer responding to a suspicious person in a strip mall parking lot by herself
• a NC officer killed during a traffic stop

D. System Concept

Based on the prior research, a system capable of detecting bullet impacts must be designed. The system will need to be able to detect bullet impacts, and determine what type of impact it is. The system also needs to be able to store the bullet impact data so that paramedics are able to review impact data. It needs to also calculate the location of the bullet impact on the ballistic plate.

Figure III.D.1 – System Concept Diagram

E. Systems Research

This section deals with the research done on how the entire project could be implemented and the components required to do so. First up, the various potential sensors such as load cells,
accelerometers, and piezoelectrics are discussed, followed by the various requirements of components in order for them to fit into the overall system.

1. Load/Force Cells

A load cell is a transducer that is used to convert force into an electric signal. The conversion is indirect and happens in two stages. The strain gauge converts the stress to an electric signal. A load cell consists of four strain gauges in a Wheatstone bridge configuration. The electric signal is output in a few millivolts which must amplified using an operational amplifier before the signal can be used. The signal can be then used by a microcontroller to calculate force or impact.

There are many different types of load cells available on the market. Unfortunately they have a large price tag. One of the cheaper load cells found in the market right now is the Transcell Single Beam Cell is $150 dollars which only has a 1000 lb Capacity.

The Futek FBB300 40 Pound force sensor is a standard load sensor which priced at $90.00 per piece. It has a rated output of 2mV/V output. It is 1.25 inches in length, and 0.31 inches wide, and is fairly flat. The dimensions of the sensor would make an ideal piece to use as a sensor in our detection system. However, the major issue I see with this impact sensor is that it is only capable of measuring 40lb of force, and a typical bullet impact results in higher forces.

When evaluating the need for sensors, we will definitely need a light weight solution. After talking to a marine, he has told me that they prefer everything to be light weight, and durable. “Marines will find a way to break everything” said Lt. Michael Bailey. The sensors need to be very durable because they will need to with stand harsh weather conditions.
Load sensors do seem like a viable solution to meet our goals; however they do not seem like the practical choice. Load sensors have a hefty price tag which will significantly raise the end cost to consumer. Load sensors do not necessarily seem to be capable of measuring high amounts of load, as we would need at reasonable cost.

2. Single/dual Axis Accelerometers

Accelerometers are a simple way to measure acceleration. There are multiple different types of accelerometers available:

- Capacitive: Metal beam or Micro machined feature produces capacitance; changes in capacitance related to acceleration
- Piezoelectric: Crystals mounted to mass voltage output converted to acceleration
- Piezoresistive: micro machined whose resistance changes with acceleration
- Magnetoresistive: Material resistivity changes in presence of magnetic field.
- Heat Transfer: Location of heated mass tracked during acceleration by sensing temperature

![Figure III.E.2.1 – Texas Instruments Cost vs. Performance](image-url)
The figure above from Texas instruments shows a good balance between cost and performance which amounts to accuracy is the Surface MM Capacitive sensors. The accelerometer measures the force of gravity in g’s. Based on the orientation measurement can be affected, If the accelerometer is oriented in a vertical position it will always read at 1g because of gravity’s pull on it.

Accelerometer sensitivity will be important in measuring our bullet impact. Sensitivity of an accelerometer is a measure of how much the output of a sensor changes as the input acceleration changes. It is measured in Volts/g.

\[
Sensitivity = \frac{\Delta V_{out}}{\Delta g} = \frac{V_{out,+1g} - V_{out,-1g}}{2g}
\]

Rigid mounting is important, utilizing bolts, or double sided tape. During setup loose wires can cause false signals, so it’s important secure wires away from the accelerometer to the mounting body. Weight of the sensor should be approximately an order of magnitude less than the object being measured. A 1 degree tilt in the 0g mounting position creates an output error equivalent to a 10 degree tilt in the +1g or -1g positions. 0g is the most sensitive to mounting errors.

There are several accelerometer applications such as measuring tilt/roll, vibration, vehicle skid detection, impact detection, and feedback for active suspension control systems.

Type IIA armor protects against 8g 9x19mm Parabellum full metal jacket round nose bullets at a velocity of 373 m/s +/- 9.1 m/s, and targets struck by such bullets experience about fifty g’s. The accelerometer therefore needs to be capable of measuring at least fifty g’s if the design is to meet Type IIA expectations.
Accelerometers are a practical and affordable option for sensors which can potentially be used as a solution to our problem.

3. Piezoelectric Sensors Viability

The utilization of a piezoelectric sensor to measure velocity, force and localization seem to be the most versatile of any sensor we have contemplated with the only major issue brought up in any brainstorming session is the actual maximum resolution the piezoelectric sensor could actually identify.

It is important to know the physical capabilities of a sensor before it’s implemented in a design. For this particular application the only important features are the frequency range, maximum voltage, the relationship between voltage and size of the sensor and its mechanical strength and impact resistance. According to the Measurement Specialties Inc. technical manual on piezoelectric film it has a:

- Wide frequency range—0.001 Hz to 10GHz.
- Low acoustic impedance—close match to water, human tissue and adhesive systems.
- High voltage output—10 times higher than piezoelectric ceramics for the same force input.
- High mechanical strength and impact resistance (10^9—10^10 Pascal modulus).
- Can be fabricated into unusual designs.
- Can be glued with commercial adhesives.

A major concern when discussing what type of sensors to use, is if the sensor will be capable (high enough resolution) of measuring at the short time interval of an actual bullet impact. According to the Measurement Specialties Piezo Film Sensors Technical Manual the lower the
thickness of the piezoelectric film the higher resolution we are capable of measuring and since a 28um thick film sensor typically has a 40MHz basic half wave resonance, film of similar thickness would be adequate.

According to the same manual, piezoelectric cables are also used in a variety of applications like weigh in motion systems used for truck inspection and taxiway sensors for aircraft identification. This cable is interesting because the force that creates the electrical energy is applied in a longitudinal stress.

4. Sensor choices

The sensor which will fit best for the application is the piezoelectric film. Piezoelectric film is typically resistant to impact, and would make the best choice for an impact detection application. The sensors are responsive to any form of deflection, especially at high speeds, and plate will encounter high amounts of deflection during impact. The piezoelectric film is also simple to mount onto the steel plate and can be easily moved to any location on it. The choice of using piezoelectric film is also a cost effective choice, as it is cheaper than accelerometers which were the secondary choice.

4. Gain Stage Expected Specs

In order to purchase appropriate components, approximations of the voltages produced by our sensors when bullets impact the plate of our system were needed. The only way to make sure components will satisfy the real world requirements based off of theoretical calculations is to assume the worst case scenario when testing the system. For starters, several arrangements were considered for the test plate in our most recent meeting, including placing the plate and sensors
against a spring system, a human analog, or a close to immobile surface. Since the spring system and the human analog will have some give when the bullet strikes and will prolong the time period our system will be recording over, we will assume the test setup where the plate is up against a solid surface because this will produce the greatest slope of voltage over time from our highest voltage sensor, the piezoelectric strips.

To find an actual slew rate requirement, the time period the bullet exerts force on the sensors is also needed. For this, several assumptions are made in order to simulate a worst case scenario. First, it is assumed that the bullet will travel through a vacuum for maximum possible velocity, and that no friction occurs between any of the components so that the only forces opposing the bullet are the material's strength and the normal force of the test set-up. It is also assumed that the bullet will stop and expend its energy before passing by the sensors, which will result in the biggest spike in output voltage. Next, it is assumed that the bullet striking the plate is the highest caliber and velocity bullet used against a police officer out of the recorded statistics in the background research, the Remington .223 caliber round, traveling at its highest recorded speed of 1140 m/s at 1524 J of kinetic energy. Finally, It is assumed the sensor used to record the force of impact is our selected piezoelectric device, Measurement Specialties' LDT0-028K series.

Calculations are begun with finding out the speed of the setup in a vacuum after being hit by the bullet using the conservation of energy equation with the assumption that the test set-up will weigh 80kg:

\[
\frac{1}{2}mv^2 = \frac{1}{2}mv^2
\]

\[
1524J = \frac{1}{2} \times 80 \times v^2
\]
\[ v = 6.1725 \text{ m/s} \]

Using this information and the fact that the test plate is 6mm in width, the two remaining unknowns, the acceleration felt by the sensors and the time interval they accelerate for, can be found from the following kinematics equations:

\[ V_f = V_i + at \]

\[ 6.1725 = 0 + at \]

\[ d = V_it + \frac{1}{2}at^2 \]

\[ 0.006 = 0 + \frac{1}{2}at^2 \]

Using substitution:

\[ a = \frac{6.1725}{t} \]

\[ 0.006 = \frac{1}{2} \times 6.172t \]

\[ t = 1.944 \text{ ms} \]

\[ a = 3174.98 \text{ m/s}^2 \]

Using these two values, the gain stage operational amplifier slew rate spec can be calculated:

Slew rate minimum = 25V/1.944ms = 12.86 mV/us

5. FPGA System Requirements

Clock speed is a critical factor for this FPGA application. Bullet impact detection is a time sensitive application and requires a high system clock speed. The bullet impact will be near instantaneous and the system will have to be able to respond/capture the data during that time.
Unless, it is possible to sample at a high rate on the sensor end, and hold until the system is capable of reading the data from the sensor.

The table below lists the velocities of a 9mm parabellum bullet:

<table>
<thead>
<tr>
<th>Ballistic performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bullet weight/type</td>
</tr>
<tr>
<td>7.45 g (115.0 gr) FMJ</td>
</tr>
<tr>
<td>8.00 g (123.5 gr) FMJ</td>
</tr>
<tr>
<td>9.1 g (140 gr) FMJ</td>
</tr>
<tr>
<td>9.5 g (147 gr) JHP</td>
</tr>
<tr>
<td>7.45 g (115.0 gr) JHP</td>
</tr>
</tbody>
</table>


Table III.E.5.1 – 9mm Parabellum Ballistics

The clock speed will have to be high enough to detect a bullet impacting the sensor at the high velocities unless the sensor is one that records that an impact has occurred after the impact, rather than during the impact.

- Using the speed for slower 9mm pistols (best case scenario):
  - the bullet travels at 900 ft/sec
  - the test plates are .6” thick
  - the bullet will travel through the plate in 55.56 microseconds

- Using the speed for a AK-47 (standard 7.62x39mm):
  - the bullet travels at 2300ft/sec
the test plates are .6” thick
the bullet will travel through the plate in 21.74 microseconds

- Using the speed for a Barrett M82A2 (standard 12.7x99mm):
  the bullet travels at 3000ft/sec
  the test plates are .6” thick
  the bullet will travel through the plate in 16.67 microseconds

This means that all of the data needs to be captured within 16.67-55.56 microseconds. Using high speed parallel analog to digital converters will aid in gathering a strong sample set of the impact. The Analog Devices (ADI) AR9220, the ADC chosen for this project, is a 10 MHz (10 MSPS) ADC which has a 12 bit digital output.

F. Basic System Block Diagram

The original overall system design is shown on the next page as Image III.F.1. The sensors chosen were the piezoelectric film, and the signals being output from them will pass through a fractional gain stage which will reduce the signal down to the input range of the analog to digital converter. Each sensor has its own gain stage because multiplexing them into one ADC would require a higher sample rate. The analog to digital converters will sample the signal and pass the data along to the FPGA which will be responsible for writing the data to RAM, and then transmitting the data to a PC for post processing. The PC will process the data, and localize the impact.
Image III.F.1 – System Block Diagram
IV. System Components

The overall system design consists of several components such as the physical test stand, the FPGA logic controller, the custom designed add-on board, and triangulation calculations. These components work together to perform complete system functionality.

A. Plate Test Stand

In order to test the basic capabilities of the system, a prototype holder was designed. This holder needed to be able to hold a ballistic plate and its attached sensors in place while it was shot, and needed to last through repeated tests. Inspiration for this holder was drawn for various commercially available pistol targets.

Since the plate is 8in or 20.3cm in both height and width, .6cm in thickness, and slightly angled in the center (shown in Image IV.A.1), the frame of the holder had to be at least its height, with an added thickness for the top section that would hold the plate from sliding out (shown in Image IV.A.2). The frame sits on a base plate that ensures stability and the falling mechanism was implemented with a simple hinge, but the system needed to be reset manually.
Since the ends of the ballistic plate used are always at an angle to a flat surface (angle X in Image IV.A.3), the frame itself would either needed to be angled with the plate or have a wider slot. Because bending the frame to fit the plate would restrict the plate from deflecting and vibrating, a non-angled configuration such as that shown in Image 1.3 was used. For our 6mm plate, the frame's width W was chosen as 3cm in order to provide a deep enough slot to ensure the plate stays put, and the 18.85mm thickness T was measured using a Vernier caliper 3cm inwards towards the center of the plate from its outermost edge.
Image IV.A.3 - Wood Frame Specifications (Top View)

Originally, the prototype of the plate holder shown in Image IV.A.4 below was tested. Its components were attached using wood screws and nails. At the time, it was attached using hinges to a 14in. wide and 3/4in. thick plank cut to fit the length of the bin holding the cement serving as the base of the system. The plank itself was attached to the bin using J-hooks. The dimensions of the holder and the effects of the test on them are explained below the image.
A - This piece is a 9 and 3/4 in. wide, 10 in. tall, and 3/4 in. thick piece of wood serving as the back of the holder. This piece is also suspected of absorbing a lot of energy when the plate is struck by the bullet. After the test, this back piece was slightly bent outward towards the back, possibly confirming the suspicion. However, we cannot know for sure if the piece was perfectly straight beforehand as this was not tested.

B - These couple of pieces are 10 in. tall pillars with a base of 1.5 in by 1.5 in. with a 11/16 in. by 3/8 in. rabbit cut out of one of the corners along the height of each pillar in order to fit the plate's 8 in. width snugly between these pieces and the back plank. These pieces were protected from shrapnel impact by plates of metal attached besides the rabbits facing inwards.
C - This piece is a 9 and 3/4 in by 2 and 3/4 in piece of wood serving as a base for the plate to rest on and as an attachment point for the hinges. The piece used in the test was 3/4 in. thick, although this particular value is not important to the functionality of the system and can be adjusted in future versions.

The main failures of the holder occurred between the vertical pieces and the back plank, around the edges of the holder struck by fragments, and in the hinges. Three more holders were made in which these issues were addressed in the following ways: since pieces B came loose from the back plank, the nails present were replaced with screws and the pieces were also glued together; to prevent shrapnel from eventually digging clean through the wood of piece C, metal plates such as those on pieces B were added; stronger hinges were used in order to avoid them failing after repeated use.

B. Piezoelectric Testing

With the new plate holders finished, the testing of the piezoelectric sensors could begin. Before this test, it was now known if these sensors would work for this application at all, and if they did, what sort of voltage output would be generated. The test was performed on the ballistic plate in one of the new plate holders and a hammer as the source of impact. A single piezoelectric sensor was attached flat to the center of the back of the plate using adhesive tape, and several blows from the hammer were delivered over the center of the opposite side while the sensor output was recorded on an oscilloscope. The best result of the series is shown below in Image IV.B.1.
Image IV.B.1 – Oscilloscope Sensor Measurement

From these results, we can assume that the output from the sensors will usually be something similar to the generalized waveform shown in Image IV.B.2 below. From the multiple peak voltages, periods, and arrival times of various sensors, we can theoretically extract the location of impact, making piezoelectric sensors the best choice for use in the project.
C. Analog Application Schematic

In order to create a system capable of utilizing different forms of input sensors, a sub-system was needed so that the vest can handle varying voltage signals from different types of sensors. The subsystem settled on was made up of gain stages taking input from the ballistic plate sensors and outputting to one ADC per sensor, which would in turn output signals to a field programmable gate array that would handle the storing of data.

For this particular design there are two different input methods that this system can use: a piezoelectric film and an electrolytic microphone. With this in mind, the analog subsection has to be versatile and simple enough to accommodate different input signals. To accomplish this, a
simple non-inverting amplifier is used for its adaptability and minimalistic design process. However the use of an op-amp with a single source has its complications.

Under negative feedback configuration, an op-amp forces the negative input voltage to equal the positive input voltage. Given that the system design utilizes a single supply op-amp configuration, the virtual reference voltage is now equal to half of the rail voltage instead of the typical ground reference seen with dual supply configuration.

![Image IV.C.1 – Non-inverting Amplifier With a Vcc/2 Reference Voltage](image)

All op-amps have some sort of input offset voltage; this voltage can either add or detract to the input signal. The offset voltage range detailed in the LM6152 data sheet is approximately +/- (5mV-8mV). With this offset one should expect the output voltage to respond by adding an extra offset of +/- (.125-.2V) due to the gain of the system.

This response is confirmed from the simulations which yielded similar results. With a 50mV signal, the observed output should behave in the manner shown in the equations below:
\[ Vo = (Vin+Vos)*(R3/R4+1) + Vref \]

\[ Vo = (50mv+5mv)*(25) + 2.5 \]

\[ Vo = 1.25\sin (wt) +.125+2.5 \]

\[ Vo = 1.25\sin (wt) +2.625 \]

Image IV.C.2 - Simulation of a Single Source Op-Amp With a 2.5 Reference Voltage

D. ADC Schematic

The ADC will operate in a single ended input with a 0 to 2xVref range. For this application the analog to digital converters will operate at 0 to 5V, which is max AVDD. It will utilize the internal reference capabilities of the ADC, and the pin configuration will be based on the instructions provided in the data sheet. Refering to page 19, and figure 21, we can see that in order to have a input range of 0 to 5V, the SENSE pin must be shorted to the REFCOM pin. Vref
and VINB must be bypassed to the REFCOM pin with a 10 uF capacitor in parallel with a 0.1 uF capacitor. Regardless of using external, or internal reference, the actual voltages used by the internal circuitry will appear across CAPT and CAPB pins, and therefore it is necessary to decouple these pins with their suggested decoupling network as can be seen in figure 10. The digital and analog voltage rail will both be 5V, therefore AVDD, and DVDD are tied to the same source.

The digital output of the ADC will be tied to the male end of the FX2 connector. Pin 0 will be the clock input pin. Pin 1:12 will be for ADC 1, Pin 13:15 for ADC 2, and 26:38 will be for ADC 3, leaving only 2 pins on the FX2 connector.

Figure IV.D.1 ADC Circuit Schematic
E. Logic System Level Implementation

The logic design for this entire system is broken up into three crucial components: the Picoblaze Core, the Memory Interface, and the Data Input controller. The overall system requires input from the three ADCs, a transmit switch, a reset button, Receive pin on the serial connected, and the system clock. It provides the required signals to operate the memory, transmit serial data, and provide a 10 MHZ clock. There are also several signals which are connected to LEDs to observe the state of the machine. The system I/O requirements are seen in the figure below.

![IntegrationLevel](image)

Image IV.E.1 - Implementation Level System Symbol

The above system is composed of several smaller components which can be seen in the figure below.
The data input controller and the picoblaze core act as controllers for the memory interface. For the intended system functionality the Data input controller will have control of the memory interface while the system is triggered and until the desired data has been written to RAM. Once that task is complete it will restore control to the picoblaze. The multiplexer is not allowed to give control of the memory interface to the data input controller unless the picoblaze allows it. The picoblaze will not take control back unless the data input controller is in its hold state. Therefore, both the Picoblaze and Data input controller will be allowed to utilize the memory interface without interference from the other component.
1. Simulation and Testing

The above simulation shows that the memory interface is sitting idle until the system triggers. Once the system triggers the memory interface begins asserting the appropriate signals to write to RAM while outputting data from the FIFO buffer and incrementing the address. At this point the data input controller has control of the memory interface. The above simulation assures that the system is capable of writing the data being input from the ADCs to RAM at appropriate addresses while adhering to the timing requirements of the RAM.

F. Asynchronous Memory Interface Controller

The memory interface controller is a crucial part of the entire design. The controller is responsible for performing asynchronous reads and writes from the on board Micron SRAM. The
controller is specifically designed with the timing constraints of the Micron SRAM controller in mind, and can be adjusted to be used with other memory devices which use a similar interface.

1. The Control Logic

The controller has several inputs and outputs which are crucial to the operation of the memory device. The controller is designed to operate on a fifty megahertz clock input. The controller has simple control logic which allows it to interface to a processor core, or any kind of combinational logic capable of providing the appropriate signals. The device requires a reset, sram_read, sram_write, 23-bit address, and 16 bit data output signals in order to operate properly. An address must be provided at all times when performing a read or write. Data output need to be valid only when a SRAM write operation is being performed. Asserting ‘sram_read’
to logic high will generate data on the ‘data_in’ bus from the address provided. The truth table below highlights the control signals, and the combinations in which they must be asserted.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Control Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clock</td>
</tr>
<tr>
<td>Reset</td>
<td>XX</td>
</tr>
<tr>
<td>Read</td>
<td>50 Mhz</td>
</tr>
<tr>
<td>Write</td>
<td>50 Mhz</td>
</tr>
</tbody>
</table>

Table IV.F.1.1 – Control Logic

2. The State-machine

The memory controller is designed as simple state machine which has several states for each operation. Since the memory controller was designed with the Micron memory controller in mind, each operation consists of five states, and the controller expects a clock input of 50 Megahertz. Therefore each state lasts 20 nanoseconds and each read or write cycle lasts for 100 nanoseconds, well beyond the required threshold of an asynchronous read/write. The following state diagram highlights the states which the machine traverses through to perform a read or write.
Image IV.F.2.1 - Memory Controller State Machine

The following table shows the corresponding signals which are asserted in each state.

<table>
<thead>
<tr>
<th>State</th>
<th>WS4</th>
<th>WS3</th>
<th>WS2</th>
<th>WS1</th>
<th>WS</th>
<th>Standby</th>
<th>RS</th>
<th>RS1</th>
<th>RS2</th>
<th>RS3</th>
<th>RS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RamCS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FlashCS</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MemWR</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MemOE</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RamUB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RamLB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RamCR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Table IV.F.2.1 - Memory Controller State Output Table

During the write cycle, the chip select, Ram upper/lower byte, Ram ADV, and clock signals are asserted low first, then 20 nanoseconds later the Memory Write signal is asserted low. And stays low for 60 nanoseconds after which it is asserted high, and the data on the memory data bus is latched by the memory device. The write cycle finishes when the chip select signal is asserted high. The read cycle is a simpler implementation because all of the signals are asserted low simultaneously along with the Memory Output Enable and then data is read in after the data on the memory data bus is valid. The time for the data to become valid is 20 nanoseconds, but to assure that the data is valid the controller waits 40 nanoseconds.
3. Simulation and On-board Functionality Testing

The Memory interface controller was tested independent from the remainder of the system. In order to test the controller, a simple combinational logic parent controller was implemented. The combinational circuit takes in an 8-bit input and converts it into a 23 bit address and a 16 bit data input. It also takes in the output from the RAM and converts to a 3 bit output. Using these components a VHDL test bench was created (See Appendix A-1 for Test Bench Code). The following simulation shows the signals for a read cycle.

![Read Cycle Test Bench](image.png)

**Image IV.F.3.1 - Read Cycle Test Bench**

The above test bench is similar to the test bench that can be found in the Micron RAM data sheet, therefore it can be concluded that when the signals are asserted to the physical device the read cycle will function correctly. The figure below shows the test bench found in the Micron RAM data sheet. In this test bench simulation the memory data bus is in high impedance mode because the simulator is not necessarily capable of outputting simulated “RAM data” when requested using the signals.
The following test bench shows the write cycle for the memory controller, and the corresponding figure shows the write cycle from the data sheet.
During the write cycle it can be observed that data is being placed on the memory data bus, and then returns the data bus to a high impedance state. The write cycle in the test bench is very similar to the test bench from the data sheet. The data sheet fails to show that Memory Write Enable signal must be asserted high before chip enable is asserted high.

![Diagram of Micron RAM Write Cycle](image.png)

Image IV.F.3.4 - Write Cycle for the Micron RAM

The write and read cycles do not actually begin until sram_read/ sram_write is asserted high and then returns low because the machine is in the standby state and does not switch to the next state until the next clock cycle.

The next step in the process was to test the physical functionality of the memory device. The test components were mapped and routed to the FPGA. The high 4 bits of the switches (first four switches) on the board were used to generate an address; the lower four bits (last four switches) were used to generate data to be written to memory. The first three buttons were used for write, read, and reset respectively. Pressing the button would begin the read or write cycle. This method
seemed to provide greatest control, and allowed time to setup the address and data. The following sequence of actions was used to test the RAM, and conclude that the controller is functional.

1. Set address switches to 1h, generating address 000010h
2. Set data switches to Ah, generating 000Ah.
3. Press write button
4. Set address switches to 2h, generating address 000020h
5. Set data switches to Fh, generating 000Fh
6. Press write button
7. Set address switches to Fh, generating address 0000F0h
8. Set data switches to Eh, generating 000Eh
9. Press write button
10. Set address switches to 1h, generating address 000010h
11. Press read button
12. Observe LEDs and match up LED pattern to Ah
13. Set address switches to 2h, generating address 000020h
14. Press read button
15. Observe LEDs and match up LED pattern to Fh
16. Set address switches to Fh, generating address 0000F0h
17. Press read button
18. Observe LEDs and match up LED pattern to Eh
The above sequence of tests proved conclusive because the data which was written to the addresses were read back and output on the LEDs. The tests all concluded that the memory device, as well as the controller is functional on the board.

4. PicoBlaze Embedded Core

The PicoBlaze is an 8-bit embedded microcontroller core for any FPGA which uses a minimal amount of logic cores. The PicoBlaze package consists of the core and UART controllers. There was no actual design work done on the core or the UART controllers and were implemented as is. All credit for the PicoBlaze design is attributed to the respective designers. The PicoBlaze is being used to control the memory interface and UART interface to read and transmit data to a PC from the on-board Micron RAM. The design work which was needed however was the port multiplexer which interfaces all of the components to function with the core. The following schematic shows the PicoBlaze and its interface to the remaining components.
5. I/O Port

The I/O port is a crucial component because it allows the PicoBlaze architecture to control multiple components with three signals: port_id, read_strobe, and write_strobe. The I/O port component acts as a specialized multiplexer.
Due to the lack of I/O ports found on the embedded core, this component had to be designed. It takes the port ID and the strobe signals to determine whether to latch the data onto the port, or read the values on the port. The following table shows the port mapping used to implement the I/O port.

<table>
<thead>
<tr>
<th>PORT_ID</th>
<th>WRITE_STROBE</th>
<th>READ_STROBE</th>
<th>PORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>01h</td>
<td>1</td>
<td>0</td>
<td>Address(7 : 0)</td>
</tr>
<tr>
<td>02h</td>
<td>1</td>
<td>0</td>
<td>Address(15 : 8)</td>
</tr>
<tr>
<td>03h</td>
<td>1</td>
<td>0</td>
<td>Address(23 : 16)</td>
</tr>
<tr>
<td>04h</td>
<td>1</td>
<td>0</td>
<td>Data Out (7 : 0)</td>
</tr>
<tr>
<td>05h</td>
<td>1</td>
<td>0</td>
<td>Data Out(15 : 8)</td>
</tr>
</tbody>
</table>
Table IV.F.4.1 - Port Mapping Table

<table>
<thead>
<tr>
<th>Port ID</th>
<th>Bits 7:0</th>
<th>Bits 15:8</th>
<th>Bits 23:16</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>04h</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Data In (7 : 0)</td>
</tr>
<tr>
<td>05h</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Data In (15 : 8)</td>
</tr>
<tr>
<td>06h</td>
<td>1</td>
<td>0</td>
<td></td>
<td>UART TX(7 : 0)</td>
</tr>
<tr>
<td>0Ch</td>
<td>0</td>
<td>1</td>
<td></td>
<td>UART RX(7 : 0)</td>
</tr>
<tr>
<td>09h</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Trig_Address(7 :0)</td>
</tr>
<tr>
<td>0Ah</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Trig_Address(15:8)</td>
</tr>
<tr>
<td>0Bh</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Trig_Address(23:16)</td>
</tr>
<tr>
<td>10h</td>
<td>0</td>
<td>1</td>
<td></td>
<td>Sram_read</td>
</tr>
<tr>
<td>10h</td>
<td>1</td>
<td>0</td>
<td></td>
<td>Sram_write</td>
</tr>
<tr>
<td>0Fh</td>
<td>1</td>
<td>0</td>
<td></td>
<td>Reset Port</td>
</tr>
<tr>
<td>0Dh</td>
<td>1</td>
<td>0</td>
<td></td>
<td>General I/O</td>
</tr>
<tr>
<td>12h</td>
<td>1</td>
<td>0</td>
<td></td>
<td>Write Port</td>
</tr>
</tbody>
</table>

The port IDs were arbitrarily chosen, and there is no special meaning behind them. Due to the 8-bit nature of the processor, the I/O port must use three separate ports and latch the values to their 8 bit slots to generate the 24 bit address, and two ports each to latch the output data and the input data. The components also require read/write signals which are generated simultaneously on the same port ID for the UART components, however due to the nature of the memory controller the memory read and write are implemented on different ports. The data and addresses must be setup and be valid before a read or write operation can occur.
6. UART & Clock Divider

The UART controllers included in the PicoBlaze kit are very simple to implement. They each have a 8 bit, 16 Byte FIFO buffers. In the transmitter, data is written to the buffer by the microcontroller and then transmitted each byte at a time. The transmitter component asserts a “buffer full” signal once the FIFO buffer fills up, and the processor must be stalled until a slot in the buffer opens for another byte to be written.

The transmitter and receiver components function on a divided clock which must be calculated to match the desired baud rate. The components require that the clock input be 16 times the desired baud rate. The clock is divided by a counter, and every time the counter reaches the calculated integer it outputs a clock high pulse to the components. The following calculations show how to calculate the integer needed.

Desired baud rate: 38,400Hz

\[
\text{baud}_{\text{count}} = \frac{\text{System Clock}}{16 \times \text{Baud Rate}_{\text{desired}}} = \frac{50\text{MHz}}{16 \times 38400\text{Hz}} = 81.38
\]

The value is required to be an integer and not a floating point value, therefore the nearest integer value of 81 is used. Therefore, the actual baud rate is:

\[
\text{Baud Rate}_{\text{Actual}} = \frac{\text{System Clock}}{16 \times \text{Baud}_{\text{Count}}} = \frac{50\text{MHz}}{16 \times 81} = 38,580\text{Hz}
\]

The actual baud rate by using a baud count value of 81 is within 0.3% error and should not be of issue. Anything within 1% error will allow room for inaccurate clock rates and poor switching times on the antiquated RS232 protocol.
7. The Assembly Programming

The program is a simple program consisting of three functions of which two are necessary for this application. The program’s goal is to read from memory using the memory interface, and then write to the UART transmit buffer. The program checks if the buffer is full before writing to the UART buffer, therefore it prevents the program from reading the memory faster than it can transmit. The program uses several registers. It uses three registers to store the 24 bit address, two registers to store the upper and lower byte of data, and a buffer register to check the UART buffer. The program will output an address before each read or write, and output the data before a write.

![Image IV.F.7.1 - PicoBlaze Program Flow Chart](image)

8. Read Function

The purpose of the read function is to operate the memory interface controller and receive data from the RAM and store into a register. The function performs the following steps:
1. Output 24 bit address to ports: address1_port, address2_port, address3_port
2. Write to the read port initiating the read operation on the memory interface
3. Read in the upper and lower bytes from memory
4. Increment the address by 1 bit.

Once the read is complete, the data is stored to two data registers and returns to the main program for the transmit function to perform the next step.

9. Transmit Function

The purpose of the transmit function is to write to the UART buffer after verifying that there is room for data to be written. The function performs the following steps:

1. Check if the UART buffer is full by comparing the UART_BUFFER register and 01h.
2. If it is full read the UART Buffer status until it is writeable, then write the lower byte
3. Check the UART buffer again, and write upper byte if its writeable

Once both the upper and lower bytes are written then the function returns to the main program and the loop is started all over again.

10. Write Function

The write function writes whatever data is present in the upper byte and lower byte registers. Therefore, data must be setup on those two registers before they are written to memory. The write function performs the following steps:

1. Output address on address ports
2. Output data upper byte to the upper byte port
3. Output data lower byte to the lower byte port
4. Write data to the write port asserting a high on the memory interface and starting a write cycle.

The write function will not be used for the primary purpose of the overall system. It was written for testing purposes to verify the functionality of the ram.

11. Simulation and Functional Testing

The test setup is a simple setup. The Nexys2 board is powered by a USB Port, and the RS232 port is connected to a USB to Serial adapter. PuTTY is then being used to monitor the COM port which the serial port is connected to. A test bench of the overall system is generated which shows the memory interface being utilized by the read and write functions, and the serial port being utilized by the PicoBlaze.

Image IV.F.11.1 - VHDL Test bench
The above test bench shows that there are five writes being performed, and then they are being “read” by five read cycles. Because the test bench is incapable of “reading” from RAM the data is not necessarily accurate, but provides foresight as to what to expect the physical device to do.

In order to test the physical device data was written to memory using the write function, and then read back, as seen in the test bench above. The sequence which was written to memory was:

<table>
<thead>
<tr>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000h</td>
<td>5057h</td>
</tr>
<tr>
<td>000002h</td>
<td>2049h</td>
</tr>
<tr>
<td>000004h</td>
<td>5552h</td>
</tr>
<tr>
<td>000006h</td>
<td>454Ch</td>
</tr>
<tr>
<td>000008h</td>
<td>2053h</td>
</tr>
</tbody>
</table>

Table IV.F.11.1 - Memory layout after write completed

The data was then read back and transmitted to the computer the Nexys 2 board is connected to generating the following results:
Image IV.F.11.2 - PuTTY Window After Doing a Write and Read Using the PicoBlaze System

From the above result it can be concluded that the system is functional and is timed properly to perform read and write functions with the RAM as well as perform transmit functions with the UART.

G. Data Input Controller

The purpose of the data input controller is to control the flow of incoming ADC data to RAM. It is composed of several parts: a clock multiplier, three 12 bit FIFO (first in first out) buffers, a clock divider, a multiplexer, an address counter, trigger, and write controller. The schematic below shows the system, and how each component is connected to the other.
The data input controller has several inputs/outputs at its highest level. A clock input is necessary to run the system. A reset signal is necessary to initialize the states of the numerous state machines. It also requires a SRAM Ready input signal which alerts the controller that the RAM is in idle state and ready for another operation. The controller also requires that there is input from the 3 ADCs via the FX-Hirose connector. The controller outputs an SRAM write signal which controls the memory interface, and when it will write to memory. It also outputs a “triggered” signal alerting the higher level system that the data input controller is operating and writing incoming data to memory. It also outputs a trigger address giving the PicoBlaze microcontroller the starting address.
1. Clock Multiplier

The clock multiplier is designated as “CLOCK” on the system schematic. It implements the internal DCM of the Xilinx FPGA and multiplies the clock from 50 MHZ to 100 MHZ. There are several reasons why the clock multiplier must be implemented of which include its primary purpose of providing a clock for the numerous FIFO buffers as well as a clock for the clock divider.

2. Clock Divider

The clock divider, designated as “ClockWrite” on the above schematic, serves two functions. Its primary function is to provide a 10MHZ sample clock to the analog to digital converters. Its second purpose is to generate a write enable signal for the three FIFO buffers. The write enable signal is asserted high for 10 ns every 100 ns. There is a delay of 30 ns from when the 10 MHZ clock is asserted high to when the write enable signal because that is the minimum delay necessary for the ADCs to process and generate new sample data.

3. Address Counter

The address counter is a simple 23 bit counter which is incremented every time the state controller reaches state four. Once it reaches its max value it will clear itself and start counting over again. The address counter is crucial to the system as it determines where the data will be written. Typically every time the read counter on the state controller FIFO buffer is incremented so is the address on the address counter.
4. FIFO Buffer

The FIFO buffers serve a very important purpose, as they allow the system to generate a queue of incoming data to write to RAM. Due to the constrains of an asynchronous write to RAM, the data being input will be at a much higher rate than the data being written. Therefore, the queue is necessary in order to prevent any data from being lost during the write process. A queue data structure (FIFO) is fairly simple, the data which entered the system first must be read first, and the data which has entered the system last will be read last.

The FIFO buffer was generated using the IPCore generator and utilizes the internal BRAM on the FPGA. It has a fairly simple operation, as it requires a 100 MHZ clock, a write, a read, and valid data to be written. When the write signal is asserted high, the data is latched into the buffer at the bottom of the stack. When the read signal is asserted high the data at the top of the stack will be read. The table below shows the operation of the FIFO buffer.

<table>
<thead>
<tr>
<th>Data In</th>
<th>Write</th>
<th>Read</th>
<th>Read Pointer</th>
<th>Write Pointer</th>
<th>Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>1</td>
<td>0</td>
<td>RP &lt;= RP</td>
<td>WP &lt;= WP+1</td>
<td>DATA[RP]</td>
</tr>
<tr>
<td>Valid</td>
<td>0</td>
<td>1</td>
<td>RP &lt;= RP+1</td>
<td>WP &lt;= WP</td>
<td>DATA[RP]</td>
</tr>
</tbody>
</table>

Table IV.G.4.1 - FIFO Buffer Truth table

The buffer when reset has both of its read and write pointers pointing to the top of the stack, and when a write signal is asserted it will write to the stack and increment the pointer to the next spot on the stack. It will also read from the stack and increment the read pointer when a read signal is asserted. When the read pointer and write pointer are equal to each other, the buffer is empty, and when the write pointer is equal to read pointer minus one spot the buffer is full.
5. Control Multiplexer

The purpose of the control multiplexer is to control reading from the three FIFO buffers, appending the ADC number, and writing the data to state_controller which in turn will write the data into RAM. The multiplexer remains in idle mode until there is data present in the ADC FIFO buffers. When the data becomes available, it begins to read from each buffer and writes it to the buffer inside the state controller component. The controller will assert a read signal to each buffer after it has read the previous value. The state diagram below shows the functionality of the control multiplexer.

![State Diagram for the Control Multiplexer](image)

Image IV.G.5.1 - State Diagram for the Control Multiplexer

The control multiplexer traverses through four states, and changes states at each clock cycle with the exception of the transition from state one to state two. In each state the multiplexer outputs data from a respective ADC buffer, and writes to the state controller. It has an idle state so that it is not writing data to the state controller when there is nothing to write.
6. State Controller

The state controller is a combination of two components, a FIFO buffer and a state machine. The state machine reads from the FIFO buffer and writes it to SRAM when the system is in triggered mode. Otherwise, if it is not triggered, and the buffer has filled up to the 2 Kb mark which is roughly 1/4th of the buffer size it will increment the read pointer of the buffer by asserting a read signal to the buffer. The state diagram below shows the functionality of the controller.

![State Controller State Diagram](image)

Image IV.G.6.1 - State Controller State Diagram

The state machine remains in state one while it is waiting for the system to trigger, and while it is in state one the buffer is filling up. When the buffer becomes 1/4th full the FIFO asserts the half_full(programmed full) signal and sets the system to state two. When the system enters state two it will continuously read from the top of the buffer and discard the data because it is old data that really does not fit in the time frame that is needed. Once the system triggers the state machine will allow the buffer to fill past the programmed full point and continuously write to
RAM and read from the buffer as the memory interface becomes available to perform an operation. The “ram_ready” signal is crucial to the system operation, if the memory interface controller fails to become available the whole system will freeze and there will no longer be any data written to the RAM. Once the system reaches state four it will also assert an “increment” signal so that the address counter increments and the next piece of data is written to the next address in RAM.

7. Trigger Controller

The purpose of the trigger controller is to initiate the write to RAM. The system will trigger once the most significant bit of one of the three ADCs has a high output, representing that something strong enough has impacted the plate to begin recording of the data which is present in the buffer to RAM. The trigger will output two signals, a “Trigger” and “Ready” signal. When the “Trigger” signal is asserted high it will send the picoblaze into a loop assuring that it does not perform any operations on the memory controller or execute any further instructions, as well as take full control of the memory interface. The “ready” signal will assert high once it triggers and enable the State controller which will begin asserting signals to the memory interface to write the data which is on the data bus to the address on the address bus.
In state one the trigger will continuously check the incoming data until it finds a byte of data where bit number 11 is high in which case it will trigger. Once it triggers it enters state two and asserts a high on both ready and trigger. The trigger will then sample the address bus and check if the address at which it triggered and the current address are equal. Once they are equal the trigger will enter state three in which it will assert all of its signals to low and transfer control of the memory interface controller to the PicoBlaze.
8. Simulation and Testing

![Simulation and Testing Image](image.png)

Image IV.G.8.1 - Data Input controller Simulation

Simulation of the data controller will be the optimal method of testing, as the whole system needs to be functional in order to do any physical testing. For the purposes of this test, the three ADC inputs are incrementing data from 000h to FFFh. For observation purposes, the point at which the system triggers is the most crucial, as the data being output from this point on is system critical data. Firstly notice that the data being output on the data out bus is not the same as the data being input, which demonstrates that the FIFO buffers are working, and are able to store roughly 300 samples prior to the trigger point for each ADC. Secondly, the address is being incremented each time there is an SRAM write which is important to ensure that there is no data which is being overwritten. Prior to the trigger point the data bus is rapidly switching data and then pausing and repeating itself because at this point the data buffer has reached the
designated point and it is removing unwanted samples. It is also outputting the 10 Mhz clock which will be routed to the sample clocks on the ADCs.

It can be concluded that the data input controller is working as intended because of these reasons:

- Address and Data are being simultaneously incremented
- Trigger address is being set to the last known address, so the picoblaze will begin reading from that address.
- It switches into triggered mode when DataIn(11) reaches high
- It is removing unwanted data at the designated mark
- It begins asserting the proper signals to SRAM once it triggers

H. Printed Circuit Board Design

In order to bring the schematic design to a physical implementation, a PCB was created. In a PCB there are many things that must be taken into account, from user specifications, part restrictions as well as manufacturing limitations.

1. Definitions

To effectively utilize the Ultiboard program, there are a few technical words that should be defined to facilitate the creation of any PCB. These are:
Via - a conductive hole that connects two layers of a PCB. By manufacturer design these holes are automatically given solder pads on each side.

Solder mask - a lacquer-like layer of polymer that resist solder which decreases the chances of stray solder, that could cause shorts in the PCB.

Solder silkscreen - a physical dawning that is used to assure that the component is placed in the correct orientation.

Board outline - an outline of outer limits of the PCB.

2. Custom Parts

There are several integral components that must be placed on the PCB. In order to place these components solder pads must be created of each unique part package. In order to create these custom part outlines the Ultiboard part wizard was used. This wizard has features that can determine the technology, package type, package dimensions, 3d color settings, pad type and dimensions, number of pins and pad numbering. The combination of the Ultiboard wizard and the mechanical information found in the particular component datasheet is enough to create solder pads for any component. Below are examples of all of the types of components used and their corresponding solder pad. The wizard is the most useful method of creating a custom solder pad but it's not the only way. If the component is not found in the Ultiboard databases, the user must create the component manually. This was the case for the 100-pin FX-2 Hirose connector, which was used to attach the PCB and the Xilinx board together. In order to properly fit the 100-pin connector, the X and Y coordinate for each pin was calculated and recorded. With the coordinate of each pin found, a "via" was placed in the corresponding coordinate. With the via's in place the custom component need to attached to the traces on the Top Copper layer.
3. PCB Layers

There are several layers that compose every PCB design, these layers range from copper layers to solder mask layers. Below is a description on each layer, its importance and how it was implemented in this design.
a. Copper Top:

This layer shows the top traces of copper used for both the analog side and the digital aspect of this project. These traces were created with the use of "place line" command with the top layer selected. Keep in mind that line dimensions can be changed where needed. In order to reduce noise a large ground area was created with the use of the "place shape" command. With this command one is capable of creating different shapes for circles to rounded rectangles.

![Image IV.H.3.a.1 - Top Copper layer](image)

b. Copper Bottom:

This layer was used to circumvent any dead ends found in the top copper layer. If you superimpose the Top Copper layer and the Bottom Copper layer, you would find there are via's that connect the top Copper and the Bottom Copper layer. If these via's are not created there would be no direct connections between layers. The use of via's and multiple layers of copper allows the designer to weave a trace through other traces, which makes it easier to connect nodes.
in the designing process. This type of connection also allows for a single cohesive and flowing circuit. The blue circles seen in both the Top Copper layer and the Bottom Copper layer are via's. In reality the blue circles represent the exposed copper in which one can solder to, connect traces, and or use as test points.

![Image IV.H.3.b.1 - Bottom Copper layer](image)

c. Silkscreen Top

In this layer a silkscreen is placed where each component is needed. This layer is used to visually locate and place component parts in the correct orientation.
d. Solder Mask Top and Bottom

Solder masks are used to prevent shorts between copper traces, as well as solder melting where it is not supposed to (stray copper). However, when Ultiboard is used, the entire board is filled with solder mask. It is up to the user to decide the location of exposed copper. In the case of both the top and the bottom layers of solder mask are used to expose the annular rings for the soldering holes as well as the pads for all surface mount components. Regular holes however do not have solder mask exposed by default. If there was any need to manually create exposed copper then, one can use the "place shape command" and or the "place line command" with either solder mask layer selected. Solder mask relief is an important element to keep in mind while creating a PCB, as the manufacture will reject an order due to it. There is a feature that allows the user to globally add solder mask relief to all components. This feature is found in the Gerber file setting. Gerber files are a type of format many PCB manufacturers use.
Image IV.H.3.d.1 - Solder masks Top

Image IV.H.3.d.2 - Solder Mask Bottom
e. Gerber Files and Manufacturing Limitations

In order to manufacture the PCB, it needs to first pass the design rules checker. The DRC is a feature embedded in the Ultiboard program which allows the user to set the manufacturing limitations straight into the program. These manufacturing limitations range from hole size to trace distances and are important because the manufacturer will not process the project until these errors are fixed. Once the PCB passes the user created DRC, manufacture specific PCB files must be exported. The manufacturer for this particular board was Advance Circuits, they required Gerber files and NCdrill files to place a PCB order. The manufacturer for this PCB also has a company based DRC called FreeDFM which can be used to check the files before placing the order. Their DRC automatically fixes some common mistakes and shows the user "show stoppers".

Image IV.H.3.d.3 – PCB Layer Compilation
I. Localization Calculations

The data from multiple sensors is useless by itself and needs to be analyzed. Since the main remaining goal of the project was to identify the location of an impact on the plate, data analysis was focused on this. All attempts to find a way to calculate the exact position of an impact on the ballistic plate from the information given by three sensors stemmed from the following basic trilateration equations:

\[ R_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2} \]

\[ R_2 = \sqrt{(x-x_2)^2 + (y-y_2)^2} \]

\[ R_3 = \sqrt{(x-x_3)^2 + (y-y_3)^2} \]

where \( R_1, R_2, \) and \( R_3 \) are the radii from the impact point to each of the three sensors, \( x_1, y_1, x_2, y_2, x_3, \) and \( y_3 \) are the locations of the sensors, and \( x \) and \( y \) are the coordinates of the impact point that need to be solved for.

The original attempt to solve this problem started by assigning each sensor a position based on known dimensions, that is, sensor 1 was located in the top left of the plate, making \( x_1, y_1 \) equal to 0, \( h \), where \( h \) is the height of the plate, sensor two was located in the top right of the plate, making \( x_2, y_2 \) equal to \( w, h \), where \( w \) is the width of the plate, and sensor 3 was located in the middle of the bottom of the plate, making \( x_3, y_3 \) equal to \( w/2, 0 \). Afterwards, \( x \) and \( y \) can be solved for using the three resulting equations and result in:

\[ x = \frac{(R_1^2 - R_2^2 + w^2)}{2w} \]

\[ y = \frac{(2R_3^2 + R_2^2 - R_1^2 + 2h^2 + 3/2w^2)}{4h} \]
This method runs into a couple of issues, however. The first stems from the fact that R1, R2, and R3 cannot be directly measured. They are each formed by the measured arrival time of the signal to a sensor minus the impact time over the speed of the wave through the plate, and while the speed of the wave can be measured, the impact time is another unknown that must be solved for. Additionally, there might be some error in the arrival times, which this method does not account for.

Due to these issues, the next attempt to solve for x and y used matrices to go through guesses for the radii compared to their actual values in a somewhat standard approach to trilateration. This method started off by adding the additional condition to the original radius equation of each radius being equal to the velocity of the impact wave through the plate multiplied by the difference between the time of recording the wave of each sensor, t1, t2, and t3, and the unknown time of impact t0. This addition is valid because both the original equations and the added condition are valid ways of calculating a radius between the two points. The appended equations are as follows:

\[
R_1 = \sqrt{(x-x_1)^2 + (y-y_1)^2} = v*(t_1-t_0)
\]

\[
R_2 = \sqrt{(x-x_2)^2 + (y-y_2)^2} = v*(t_2-t_0)
\]

\[
R_3 = \sqrt{(x-x_3)^2 + (y-y_3)^2} = v*(t_3-t_0)
\]

The next step was to take the Taylor Series transforms of R1, R2, and R3 about an estimation point xe, ye in order to make the equations more matrix friendly and to incorporate the estimation point for use in the iterative process of this method. The first two terms of the transforms about xe, ye are:
R1 = sqrt((xe-x1)^2 + (ye-y1)^2) + (xe-x)*(xe-x1)/R1 + (ye-y)*(ye-y1)/R1

R2 = sqrt((xe-x2)^2 + (ye-y2)^2) + (xe-x)*(xe-x2)/R2 + (ye-y)*(ye-y2)/R2

R3 = sqrt((xe-x3)^2 + (ye-y3)^2) + (xe-x)*(xe-x3)/R3 + (ye-y)*(ye-y3)/R3

Since these are Taylor Series transforms, these equations are still equal to the velocity multiplied by the time difference discussed earlier, and the terms xe-x and ye-y are the error terms Δx and Δy that will be solved for. However, that means they still contain the unknown t0, but that term can be canceled out by subtracting R3 from R1 and R2 from R1 as follows:

R1 – R2 = v*(t1-t2) + sqrt((xe-x2)^2 + (ye-y2)^2) - sqrt((xe-x1)^2 + (ye-y1)^2) =

= Δx*((xe-x1)/R1 – (xe-y2)/R2) + Δy*((ye-y1)/R1 – (ye-y2)/R2)

R1 – R3 = v*(t1-t3) + sqrt((xe-x3)^2 + (ye-y3)^2) - sqrt((xe-x1)^2 + (ye-y1)^2) =

= Δx*((xe-x1)/R1 – (xe-y3)/R3) + Δy*((ye-y1)/R1 – (ye-y3)/R3)

These equations can finally be put in matrix form as such:

[v*(t1-t2) + a  v*(t1-t3) + b] = [Δx  Δy]A

Where:

a = sqrt((xe-x2)^2 + (ye-y2)^2) - sqrt((xe-x1)^2 + (ye-y1)^2)

b = sqrt((xe-x3)^2 + (ye-y3)^2) - sqrt((xe-x1)^2 + (ye-y1)^2)

And A is a 2 by 2 matrix with terms:

A1 = (xe-x1)/R1 – (xe-y2)/R2
A2 = (xe-x1)/R1 – (xe-y3)/R3

A3 = (ye-y1)/R1 – (ye-y2)/R2

A4 = (ye-y1)/R1 – (ye-y3)/R3

The central idea behind this method is that an iterative process will result in a very good approximation of the impact position by choosing an estimation point arbitrarily, inserting it into the above matrix and solving for the error terms every iteration:

\[
\begin{bmatrix}
\Delta x \\
\Delta y
\end{bmatrix} = \frac{[v*(t1-t2) + a \ v*(t1-t3) + b]}{A}
\]

Then, the resulting error terms of one iteration are used to calculate a new estimation point which is used in the next iteration. After a few iterations, as few as three or four, the resulting error terms should be small enough to consider the next calculated estimation point the actual x, y. These calculations were simulated in MATLAB using simulated dimensions and specifications, and the code for them is available in Appendix II.
V. Results

The system was first tested in a laboratory setting in which a function generator was connected to the inputs of the analog to digital converters to verify that the logic system and ADC system were working together in unison. For the purposes of this test, a 20 kHz five volt sinusoidal wave was put into the input of the ADC, triggering the system when it reached 2.6 Volts. The system began recording the data into RAM, and then the data was transferred to a computer system to be analyzed and plotted. The result of this system test was the following graph:

![Sin wave recording @ 20 kHz](image)

Figure V.1 – Result of the PSRAM Memory Transfer
Following the first test, a more realistic set-up was attempted. The sensors were directly attached to the 8in by 6in ballistic plate, which was then inserted into its holder. With the entire system assembled, the plate was struck several times using a hammer. The hammer blows were expected to produce a large enough signal from the piezoelectric sensors to trigger the ADCs and begin the recording process. However, the system only recorded the “at rest” signal from the sensors, indicating that the trigger was a false positive or that the voltage was high enough to trigger temporarily, but that the waveform died down before the recording process began. The possible problem could be in a few places, but the issue most likely lies with the system not reacting fast enough to the sensory input chosen.

Since the hammer test did not yield usable results, the localization calculations could not be tested with actual data. They were, however, tested using dummy data through the MATLAB code provided in Appendix II. The code tests if the method works for two points, 10, 5, and 7, 2, on an imaginary plate 12 by 12 units in area. The code proves that the method is a valid one, with a possible result within ~10% of actual being achieved within three to four iterations.
VI. Sources


<http://cnx.org/content/m15670/latest/>.


APPENDIX I

Objective:
Create a memory controller to input data from ADCs and store the data in the onboard PSRAM for future analysis.

Approach:
To create a PSRAM memory controller, the following sub-components had to be constructed:

- Digital Clock Manager
- Memory Interface
  - BCR Write
  - BCR Read
  - SRAM Write
  - SRAM Read
  - Burst Write
  - Burst Read
- ADC Simulator
PSRAM Memory Specifications:

Figure 1.0 – Burst Write Cycle Timing Requirements
Figure 1.1 – Burst write cycle timing diagram
Figure 1.2 – BCR write timing diagram
Figure 1.3 – Bus configuration register definition
Components:

Digital Clock Manager:

Inputs:
- RST_IN – A reset pin
- CLKin_IN – The input clock that will be modified

Outputs:
- CLKFX_OUT – The output clock
- CLKFX180_OUT – The output clock, shifted in phase by 180 degrees
- CLKin_IBUFG_OUT – Open output pin

The digital clock manager takes an input clock, the 50MHz square wave, and outputs a square wave of a desired frequency. The DCM outputs the clock that is used by the rest of the memory controller. The output frequency has been chosen to be 60MHz because it lies under the maximum operating frequency of the PSRAM and it is fast enough to write data from 3 ADCs simultaneously.

The DCM generates a clock that satisfies the lowest speed grade memory that could be on the Nexys2 board. The configuration of the burst configuration register for continuous mode will write 128 words before refreshing the PSRAM. With a clock frequency of 60MHz, this entire write sequence should 132 clock cycles which will take 2.2 microseconds.

ADC Simulator:
Figure 1.7 – ADC Simulator

Inputs:

- **ready** – The ready signal from the PSRAM memory controller. When this signal is high, the output is enabled. When this signal is low, the output is disabled.
- **clk** – The 60MHz clock input from the Xilinx digital clock manager.

Outputs:

- **ce** – The output to chip enable pin of the PSRAM memory controller
- **data(15:0)** – The data bus to the PSRAM memory controller to simulate output from the ADCs.

The ADC simulator component fills up an array with a pre-defined pattern. When the ready signal is high, and an array is full, the state machine iterates though the array and outputs the next word to the data bus. The ADC simulator controls the chip enable pin on the PSRAM memory controller. The chip enable pin is driven low when the ready signal is high and the ADC is about to output data to the PSRAM.

Figure 1.8 – ADC simulator test bench
PSRAM Memory Controller:

![Diagram of PSRAM memory controller](image)

Figure 1.9 – PSRAM memory controller

Inputs:

- **clk** – The clock input from the 50MHz oscillator
- **reset** – A reset input tied to a button input on the Nexys2 board
- **ready** – A ready signal input tied to a switch on the Nexys2 board
- **waits** – An input from the output of the PSRAM device to give information on the status of the...
PSRAM device
- sram_read – An input from a button on the Nexys2 board to tell the PSRAM device to do an asynchronous read
- sram_write - An input from a button on the Nexys2 board to tell the PSRAM device to do an asynchronous write
- aswitch – An input from a switch on the Nexys2 board to tell the PSRAM device to switch from asynchronous mode to synchronous burst mode
- bcr_write - An input from a button on the Nexys2 board to tell the PSRAM device to do an asynchronous write to the Bus Configuration Register
- bcr_read - An input from a button on the Nexys2 board to tell the PSRAM device to do an asynchronous read from the Bus Configuration Register
- adata_out – Inputs from the switches on the Nexys2 Board to dictate what to write during an asynchronous write

Outputs:
- Memory Control Logic – adv, lb, ub, oe, we, ce, cre – Controls PSRAM
- MemDB_reg(15:0) – disconnected legacy port
- addr(22:0) – Outputs the address to the PSRAM address bus
- data(15:0) – Outputs data to the PSRAM data bus
- led(7:0) – Debugging LEDs
- ce_disable – The out to disable the Flash device that shares the address and data buses with the PSRAM device
- clk2 – A 60MHz output clock

---

Figure 1.10 – PSRAM burst write test bench

The PSRAM controller is the component that contains the ADC simulator, BCR write component, and the Xilinx Digital Clock Manager. Continuous burst mode just continually writes to the sequential column in the row of memory until it reaches the end of the row. The rows of the MT45W8MW16 contain 128 words, and if we operate the memory controller at 60MHz then we are writing data for 2.13 microseconds and the total time spent writing the row of data is 2.2 microseconds which means that there will be 4 clock cycles where we do not write data to memory because of the row boundary crossing functionality in
CellularRAM. A implementation idea is 2 arrays of 128 std_logic_vectors(15 downto 0) and we poll the ADCs every 100 nanoseconds and store the data sequentially in the array and when one of the arrays fills up we burst all of the data in the array to the PSRAM. It will take the buffer around 12.8/n (n being the number of ADCs) microseconds to fill up and 2.2 microseconds to dump the information into PSRAM. When we are bursting data from one of the arrays to PSRAM, we continue to poll the ADCs and store the values in the second array, and continue to alternate arrays until the information from the ADCs is useless.

Figure – 1.11 – SRAM_Controller2 internal view
Memory Interface:

The memory interface holds the core state machines that allow the system to do asynchronous reads, asynchronous writes, burst reads, burst writes, BCR reads, and BCR writes.

![Memory Interface Diagram](image)

**Figure 1.12 – Memory Interface**

Inputs:

- clk – 60MHz input clock
- clk2 – 60 MHz input clock (180 degrees phase shift of clk)
- reset – reset pin to reset state machine
- sram_read - An input from a button on the Nexys2 board to start the asynchronous read state machine
- **sram_write** - An input from a button on the Nexys2 board to start the asynchronous write state machine
- **bcr_read** - An input from a button on the Nexys2 board to start the bcr asynchronous read state machine
- **bcr_write** - An input from a button on the Nexys2 board to start the bcr asynchronous write state machine
- **waits** – An input from memory asserted when it is refreshing
- **burst_switch** – A input from a switch on the Nexys2 board to start the burst write state machine
- **ces_reg** – An input from the ADC simulator telling it that the buffers have started to fill up
- **address(22:0)** – an address input to dictate where to start an asynchronous read or write, as well as a synchronous read or write
- **data_out(7:0)** – An input from the switches on the Nexys2 board to dictate what to write in asynchronous mode
- **data_input(15:0)** – Data input from the ADC simulator

### Outputs

- **Memory Control Logic** - bcr_loaded, RamCS, FlashCS, MemWR, MemOE, RamUB, RamLB, RamCRE, RamADV, RamClk
- **Data_in(15:0)** – Data output read by sream_controller2
- **Ledout(2:0)** – Diagnostic LED output
- **MemAdr(22:0)** – 23-bit memory address bus
- **MemDB(15:0)** – 16-bit memory data bus
Figure 1.13 – BCR Write
Figure 1.14 – Load internal buffers
Figure 1.15 – Initialize Burst Write
The Memory interface simply monitors for buttons for the asynchronous reads and writes, a reset switch, a burst ready switch, and a ready signal from the ADC simulator. Figures 1.13, 1.14, 1.15, and 1.16 show details of what the memory interface accomplishes. In figure 1.13, the first process occurs with a button push, an asynchronous write occurs with slight modifications. The address bus is specified a value that will be written to the BCR and the cre pin is asserted high. Figure 1.14 shows what happens when burst_switch is enabled. When burst_switch is enabled, the ADC simulator starts storing information in its internal buffers, then when an internal buffer is filled ces_reg goes low and a burst write process begins. Figure 5 shows a closer look at the initialization of a burst write. After the system is done burst writing to memory, then the contents of the memory can be verified with a manual asynchronous read from a desired address.
Process:

The following is the process for the complete burst write to memory:

- BCR Write
- Toggle Burst_Switch
- Fill internal buffers with input from ADCs
- Burst write information from buffers to memory
- Asynchronously read back the information stored into memory

The first step in this process is to use the configuration register to configure the BCR. The BCR is configured by applying the desired binary sequence on the 23-bit address line and then doing an asynchronous write with the CRE pin high. This process will overwrite the BCR, RCR, or DIDR depending on what address bits 18 and 19 are set to. The following attributes were chosen to configure the BCR:

- Select BCR register
- Synchronous mode
- Fixed latency
- Latency code 3
- Wait active low
- Wait asserted during delay
- Full drive strength
- No wrap
- Continuous burst mode

By selecting all of these options, the address needed to be set to “0001000101100000011111”. Selecting the BCR register, and synchronous mode sets the memory to be ready for a burst write or a burst read. The latency mode and the latency code are used instead of monitoring the wait pin back from memory. Using fixed latency, the data will start writing after a desired amount of clock cycles from the CE and he ADV pin going low. A fixed latency has been chosen over a variable latency for the same reason, avoiding the monitoring of the wait pin. Wait is active low which means that just before it starts writing data to memory the wait pin will be asserted low for 3 clock cycles before writing data. Full drive strength was chosen because power is not as issue in this problem. No wrap was selected because it is the way the end of the row writing was designed. Continuous burst mode was chosen to get the data out of the buffers as fast as possible, in continuous burst mode data is written until the end of the row of memory (up to 128 words).

The second step is simply waiting for an external switch to be set so that the buffers could start filling up. The third step is then to fill up two internal buffers with data from the ADCs, these buffers are the size of 128 words so that it can write an entire row into memory. While an individual buffer was being dumped into memory, a second buffer is storing information from the ADCs so that data is not lost, and then when that second buffer fills up it is dumped into memory while the first starts to fill again. The burst write process is identified in figure 1.1 and shows which signals to toggle at which times. Once the data is
written into memory, an asynchronous read can then be done to verify the contents of a specific memory address.

**Project Issues Using Burst Write:**

Due to a lack of time, burst write is not implemented in our final design. Our final design includes an asynchronous system and several fifo buffers. The system was able to write synchronously to memory but when the data was read back it did not match expected results. There were several issues that needed to be focused on. The largest issue that needed attention was the possibility of failure implementing fixed latency and specifying a latency code instead of using variable latency and monitoring the wait signal. Verifying the correct synthesis of the ADC simulator and making sure that the data was input at the correct times into the memory interface was also an important debugging step that we did not have time to complete.
APPENDIX II

%MQP Trilateration Simulation

For the purposes of this simulation, we assume a plate 12cm by 12cm will be used. On it, there are three sensors at locations (1,11), (11,11), and (6,1), that are capable of detecting the exact time at which a wave travelling from an impact point strikes them. The time of impact, t₀, is unknown; therefore each sensor n, where n=1, 2 or 3, can only detect times tₙ-t₀.

We also assume the speed of said wave in this plate is 10000 cm/s and that the impact happens at point (10,5). This value is never used in the simulation calculations but must be solved for, and is only used to calculate the results that the sensors would generate:

\[ v = 10000 \]

\[ R_1 = \sqrt{9^2 + 6^2} = 10.817 \]
\[ R_2 = \sqrt{1^2 + 6^2} = 6.083 \]
\[ R_3 = \sqrt{4^2 + 4^2} = 5.657 \]

\[ t_1-t_0 = \frac{10.817}{v} = .0010817 = 1.082 \text{ ms} \]
\[ t_2-t_0 = \frac{6.083}{v} = .0006083 = .608 \text{ ms} \]
\[ t_3-t_0 = \frac{5.657}{v} = .0005657 = .566 \text{ ms} \]

Assuming impact happens at some time t₀=5ms, the sensors would return the following data:

\[ s_1 - 6.082\text{ms} \]
\[ s_2 - 5.608\text{ms} \]
\[ s_3 - 5.566\text{ms} \]

Other than the predetermined locations of the sensors, this is the only information the below simulation can use to find the impact point.

%Defining sensor locations

global x1 x2 x3 y1 y2 y3

\[ x_1 = 1 \]
\[ y_1 = 11 \]

\[ x_2 = 11 \]
\[ y_2 = 11 \]

\[ x_3 = 6 \]
\[ y_3 = 1 \]
% Defining sensor results and speed of wave

```matlab
global t1 t2 t3 v

t1 = .006082
t2 = .005608
t3 = .005566

v = 10000
```

% Selecting the original estimation point in the center of the plate

```matlab
global xe1 ye1

xe1 = 6
ye1 = 6
```

% Start of first iteration

```matlab
global R1 R2 R3 dx1 dy1

R11 = sqrt(((xe1-x1)^2)+((ye1-y1)^2))
R21 = sqrt(((xe1-x2)^2)+((ye1-y2)^2))
R31 = sqrt(((xe1-x3)^2)+((ye1-y3)^2))

T1 = [0 0]
T1(1) = (v*(t1-t2))+sqrt(((xe1-x2)^2)+((ye1-y2)^2))-sqrt(((xe1-x1)^2)+((ye1-y1)^2))
T1(2) = (v*(t1-t3))+sqrt(((xe1-x3)^2)+((ye1-y3)^2))-sqrt(((xe1-x1)^2)+((ye1-y1)^2))

A1 = [ 0 0; 0 0]
A1(1) = (((xe1-x1)/R11)-((xe1-x2)/R21))
A1(2) = (((xe1-x1)/R11)-((xe1-x3)/R31))
A1(3) = (((ye1-y1)/R11)-((ye1-y2)/R21))
A1(4) = (((ye1-y1)/R11)-((ye1-y3)/R31))

E1 = [0 0]
B1 = inv(A1)
E1 = T1*B1
dx1 = E1(1)
dy1 = E1(2)
```

% Start of second iteration

```matlab
global xe2 ye2 dx2 dy2

xe2 = xe1 + dx1
```
ye2 = ye1 + dy1

\[ R_{12} = \sqrt{((x_{e2} - x_1)^2) + ((y_{e2} - y_1)^2)} \]
\[ R_{22} = \sqrt{((x_{e2} - x_2)^2) + ((y_{e2} - y_2)^2)} \]
\[ R_{32} = \sqrt{((x_{e2} - x_3)^2) + ((y_{e2} - y_3)^2)} \]

\[ T_2 = \begin{bmatrix} 0 & 0 \end{bmatrix} \]
\[ T_{2(1)} = (v*(t_1 - t_2)) + \sqrt{((x_{e2} - x_2)^2) + ((y_{e2} - y_2)^2)} - \sqrt{((x_{e2} - x_1)^2) + ((y_{e2} - y_1)^2)} \]
\[ T_{2(2)} = (v*(t_1 - t_3)) + \sqrt{((x_{e2} - x_3)^2) + ((y_{e2} - y_3)^2)} - \sqrt{((x_{e2} - x_1)^2) + ((y_{e2} - y_1)^2)} \]

\[ A_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \]
\[ A_{2(1)} = \frac{((x_{e2} - x_1)/R_{12}) - ((x_{e2} - x_2)/R_{22})}{R_{12}} \]
\[ A_{2(2)} = \frac{((x_{e2} - x_1)/R_{12}) - ((x_{e2} - x_3)/R_{32})}{R_{32}} \]
\[ A_{2(3)} = \frac{((y_{e2} - y_1)/R_{12}) - ((y_{e2} - y_2)/R_{22})}{R_{22}} \]
\[ A_{2(4)} = \frac{((y_{e2} - y_1)/R_{12}) - ((y_{e2} - y_3)/R_{32})}{R_{32}} \]

\[ E_2 = \begin{bmatrix} 0 & 0 \end{bmatrix} \]
\[ B_2 = inv(A_2) \]
\[ E_2 = T_2*B_2 \]
\[ dx_2 = E_2(1) \]
\[ dy_2 = E_2(2) \]

%Start of third iteration

global xe3 ye3 dx3 dy3

xe3 = xe2 + dx2
ye3 = ye2 + dy2

\[ R_{13} = \sqrt{((x_{e3} - x_1)^2) + ((y_{e3} - y_1)^2)} \]
\[ R_{23} = \sqrt{((x_{e3} - x_2)^2) + ((y_{e3} - y_2)^2)} \]
\[ R_{33} = \sqrt{((x_{e3} - x_3)^2) + ((y_{e3} - y_3)^2)} \]

\[ T_3 = \begin{bmatrix} 0 & 0 \end{bmatrix} \]
\[ T_{3(1)} = (v*(t_1 - t_2)) + \sqrt{((x_{e3} - x_2)^2) + ((y_{e3} - y_2)^2)} - \sqrt{((x_{e3} - x_1)^2) + ((y_{e3} - y_1)^2)} \]
\[ T_{3(2)} = (v*(t_1 - t_3)) + \sqrt{((x_{e3} - x_3)^2) + ((y_{e3} - y_3)^2)} - \sqrt{((x_{e3} - x_1)^2) + ((y_{e3} - y_1)^2)} \]

\[ A_3 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \]
\[ A_{3(1)} = \frac{((x_{e3} - x_1)/R_{13}) - ((x_{e3} - x_2)/R_{23})}{R_{13}} \]
\[ A_{3(2)} = \frac{((x_{e3} - x_1)/R_{13}) - ((x_{e3} - x_3)/R_{33})}{R_{33}} \]
\[ A_{3(3)} = \frac{((y_{e3} - y_1)/R_{13}) - ((y_{e3} - y_2)/R_{23})}{R_{23}} \]
\[ A_{3(4)} = \frac{((y_{e3} - y_1)/R_{13}) - ((y_{e3} - y_3)/R_{33})}{R_{33}} \]

\[ E_3 = \begin{bmatrix} 0 & 0 \end{bmatrix} \]
\[ B_3 = inv(A_3) \]
\[ E_3 = T_3*B_3 \]
dx3 = E3(1)
dy3 = E3(2)

%Start of fourth iteration

global xe4 ye4 dx4 dy4

xe4 = xe3 + dx3
ye4 = ye3 + dx3

R14 = sqrt(((xe4-x1)^2)+((ye4-y1)^2))
R24 = sqrt(((xe4-x2)^2)+((ye4-y2)^2))
R34 = sqrt(((xe4-x3)^2)+((ye4-y3)^2))

T4 = [0 0]
T4(1) = (v*(t1-t2))+sqrt(((xe4-x2)^2)+((ye4-y2)^2))-sqrt(((xe4-x1)^2)+((ye4-y1)^2))
T4(2) = (v*(t1-t3))+sqrt(((xe4-x3)^2)+((ye4-y3)^2))-sqrt(((xe4-x1)^2)+((ye4-y1)^2))

A4 = [ 0 0; 0 0]
A4(1) = (((xe4-x1)/R14)-((xe4-x2)/R24))
A4(2) = (((xe4-x1)/R14)-((xe4-x3)/R34))
A4(3) = (((ye4-y1)/R14)-((ye4-y2)/R24))
A4(4) = (((ye4-y1)/R14)-((ye4-y3)/R34))

E4 = [0 0]
B4 = inv(A4)

E4 = T4*B4

dx4 = E4(1)
dy4 = E4(2)