\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Optimized KNN Implementation on an FGPA Cluster**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Proposal

Andrew Farran

Grant Fettig

Daniel McKee

Western Michigan University

ECE 4810

12/08/2017

[**Project Overview**](#_dj4dk6whsvl3) **2**

[Background and Description](#_o100kb4qwxj0) 2

[Need Statement](#_8741jf79a01w) 2

[**Project Specifications**](#_37zy72vbs5iq) **3**

[Physical Characteristics](#_r0viyp9jbl2v) 3

[Functionality](#_apf4twjfatph) 3

[Constraints](#_88j7bix7z1tx) 4

[Economic](#_tcjwyssk57r9) 4

[Health and Safety](#_h4zecmt5g255) 4

[Environmental and Sustainability](#_v9no6qmagcfn) 5

[Ethical](#_37ipe3u4ha1k) 5

[Social](#_mx82tb721iel) 5

[Manufacturability](#_ql1ze5orjzkr) 6

[Physical](#_nujl8a2kzf35) 6

[Economic](#_lb7e07iv5wy0) 6

[Resources](#_jcv68auvuor) 7

[**Design**](#_deotuivkkth4) **7**

[Literature Search](#_vbxd1hc77zdx) 7

[Concept](#_4rmx0jiovj19) 10

[Figure 1: Block Diagram of the FPGA Cluster Hardware Accelerator. The master Zybo Board will communicate to the slave Nexys 4 FPGA boards via Ethernet.](#_os5jyzrfom9n) 10

[Alternative Approaches](#_b3bb0edjjf2m) 11

[Precedence Matrix](#_n8u68845udti) 11

[The precedence matrix shown in figure 2 helps organize the different components of the entire system. By highlighting the dependencies among these components, the ideal sequence to complete each section of the project is determined.](#_1nqwsivxcqlg) 12

[Hardware Components](#_tu56oq5orcis) 12

[Software Components](#_l1qfsi2tff4a) 12

[Models and Simulations and Design Validation](#_uclrxysg9ej) 13

[Testing Procedures](#_1q5vsvq1w4q7) 13

[**Plan of Work**](#_hj27pnbukr3e) **15**

[Work Flow Diagram](#_bh1y15lk4915) 15

[Critical Path Network](#_c4rqlijpsbr4) 16

[CPN Key](#_r6kohcdostbt) 17

[**Project Deliverables**](#_788cesmgsmxt) **19**

[**Conclusions**](#_zi3xzunc3gn3) **19**

[**References**](#_yclw1rqh2lnn) **20**

# Project Overview

## Background and Description

This project is a continuation of the Hardware Accelerator created by WMU student’s Martin Cowley and Alan Irwin in 2016. A hardware accelerator uses custom hardware to solve a problem faster and more efficiently than it can be solved in software. This hardware accelerator has been developed to optimize the k-nearest neighbor machine learning algorithm. Cowley and Irwin’s project utilized a single development board with an ARM processor and a field programmable gate array (FPGA) to perform arithmetic and data operations. Martin Cowley has thus continued his work by creating a pipelined design for the FPGA logic. It is the goal of this team to build and utilize a FPGA cluster to further increase the performance of Cowley and Irwin’s Hardware Accelerator. Additionally, the FPGA cluster shall be a prototype that can be further used to accelerate multiple different algorithms in the future.

## Need Statement

Developing a Master-Slave parallel FPGA cluster design will improve upon execution time, and performance for various machine learning algorithms. This design optimization will allow more data to be analyzed in a shorter amount of time, providing more accurate predictions in weather, health, and image processing applications.

# Project Specifications

## Physical Characteristics

The physical components of the FPGA Cluster are a Zybo Z7-10 development board, three ethernet cables, an ethernet switch, and two Nexys 4 trainer boards. A PC and a USB cable will also be needed but these are external to the main focus of the project. The Zybo board has many hardware components. The hardware components being utilized in the cluster are an ARM processor, 1GB of DDR3 onboard memory, 10/100/1000 ethernet PHY, and a FPGA. The processor is a 650 MHz dual-core ARM Cortex-A9. The FPGA is equivalent to the Artix-7 FPGA with 17,600 look-up tables, 32,200 flip-flops, and 270KB of block RAM. The hardware components being used on the Nexys 4s are the Artix-7 FPGA and 10/100 ethernet PHY.

## Functionality

Before the FPGA cluster can perform its operations, the data set must be transmitted from the PC to the onboard memory of the master Zybo board via UART. After the data has all been downloaded the FPGAcluster can perform its operations. The overall task for this FPGA cluster is to obtain the kth nearest neighbors of the data set and to store these into the DRAM of the Zybo board. The first step is one core of the processor directs data to the Zybo FPGA and the two slave FPGAs. The FPGAs then find the distances between the vectors and then wait for the other processor core to ask for the data. This processor core then receives the vector distance and decides if the vector distance should be stored into the kth nearest or discarded. If the vector should be stored then the processor stores it and then sorts the kth nearest array.

## Constraints

The constraints of this project center around hardware availability and the cost of FPGA boards as well as the bit-rate of data transfer through a communication BUS. An ideal FPGA board for our design would be one with a large amount of on-board storage, but because of the higher costs of FPGA boards it may not be a financially viable option. Finding the balance between economic realizability and level of optimization needs to be considered.

## Economic

FPGA’s used to be very expensive, but as time goes on the price has dropped significantly. Using an FPGA cluster may soon be the most economical choice for running complex algorithms. Companies such as amazon have cloud based services where a user can download their hardware description language(HDL) and run their programs on an FPGA cluster. Graphical Processing Units (GPUs) are generally cheaper than FPGAs so that could be an approach to solving this problem but when considering power consumption and speedup the FPGA is more practical for the given problem [1].

## Health and Safety

Any person with general knowledge of electronic devices can use this product without harming their health. All electrical components are using standard DC voltages. A plastic case will be built around the various components mainly to protect the hardware but this will also avoid electrical shock.

## Environmental and Sustainability

To achieve the speed required to compute the kth nearest algorithm, a FPGA or GPU must be used. Researchers have programmed the k-nearest neighbors algorithm with OpenCL on GPUs and in Verilog on a FPGA and tested them against each other [2]. The algorithm on a FPGA has shown a decrease in energy consumption while the use of GPUs with this algorithm has shown the opposite despite achieving the same goal [2].

## Ethical

Advances in computing improves nearly all technology industries. Some examples are: better cancer-screening devices, more accurate systems targeted for missiles, or improved personal electronic devices. These can be used to save lives, take lives, share information, and intrude on people’s privacy. The ethical stance of using high performance computing is all in the way the user intends on applying the technology.

## Social

The types of technology that will utilize the design that the project focuses on will not be purchased by an average consumer at an electronics store. The machine learning and parallel architecture will be utilized to make predictions regarding things such as weather, and image processing for x-rays and MRIs in the medical field [3]. Thus, the social impact will be that people will benefit from the advancement of these optimizations in a secondary capacity by watching the weather or going to the doctor.

## Manufacturability

The FPGA cluster has a high manufacturability. The hardware can be purchased and the software can be easily downloaded to the different components. If the product was to be mass produced it would be cheaper to replicate the FPGA functionality with an integrated circuit as there are companies that take FPGA designs and turn them into integrated circuits. The processor could be purchased separately in the case of this design.

**Project Feasibility**

## Physical

As this is a continuation of a project from previous semesters, the design of the board connection layout as well as the software portion are physically realizable. On top of the existing design, there are numerous journal articles that have shown the physical design the group will work with is feasible. A pipelined architecture allows for the optimization of throughput as well as latency [4]. The combination of the pipelined architecture in addition to a parallelized design of the communication has shown they are both realizable as well as more efficient [5].

## Economic

This project does not require any manufacturing from the group and does not require a design to be sent to a third party for manufacturing. Everything that will be used to meet the specification requirements for this projected can be purchased. This will include the Zybo boards, the Nexys boards and well as any communication BUS that is to be decided at a later time. While these boards are not as cheap as many microcontrollers, with the Zybo boards costing around $200 each and the Nexys boards costing around $350 each, they are on the lower end of the spectrum for FPGA boards pricing. The minimum price that is expected for this project is $850 with a majority of the cost coming from the cost of FPGA boards.

## Resources

Aside from potential guidance from our project sponsor, the project is also being worked on by Martin Cowley. His current focus is the continuation of the hardware coding while the focus for the group will be to create optimized communication between the FPGA boards to fully utilize the partitioned parts of the algorithm that is spread throughout the cluster.

# **Design**

## Literature Search

The use of an FPGA along with a microcontroller allows the hardware programming design to be tailored to the application requirements [6]. Great need has been expressed for acceleration of hardware. With no locality of data, a parallel software implementation alone is not very efficient. To utilize the pipelined design of this architecture, it is important to have the data stored in a single, localized space [7]. L.W. Kim et al. have described their usage of pipelining and parallelization of hardware in the context of Artificial Neural Networks. While their algorithm will not be used by the group for this project, the concept of minimizing the complexity of data flow is essential [7].

Kilts uses Henry Ford’s assembly line to represent the idea of a pipeline [4]. As the data goes through the various hardware stages, other data begins to follow once a certain stage is reached, just like cars on an assembly line. On top of a pipelined design, Kilts discusses the essential design requirements to maximize hardware speed [4]. Included in this are: designing the hardware for high-throughput, ensuring low latency, and providing layers to separate combinational logic amongst other things [4]. Much of this can be accomplished with hardware designers adjusting the numerical precision, number of stages in a pipeline, as well as the number of cores that these processes are running on [6].

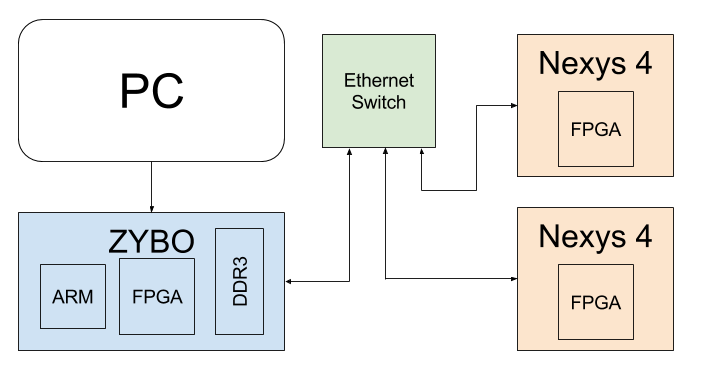
A popular learning algorithm that is instance-based is the kth-nearest neighbor [8]. This machine learning algorithm must identify and place the nearest neighbors into a query, then determine the distance to the next nearest neighbor with *k* amounts of data points [8]. Data points of this algorithm are often represented in terms of a dissimilarity function [11]. A function with larger values represent further distances from the main data point [9]. In order to determine the category of a desired element, the algorithm must take in a parameter from the result of a distance calculation [10]. This parameter is the distance between *k* data points and the main point [10]. That new point belongs to the category that it shares the closest distances with [10]. For example, while fruits are generally defined as tasting sweet, a sweet potato is categorized as a vegetable [10]. Despite its sweetness, it is closer in distance to more data points of the vegetable category such as green beans, lettuce and various types of potatoes [10].

In a more hardware driven sense, researchers have implemented FPGA clusters, which showed significant speed-up over software-only solutions. One implementation of a k-means algorithm on a FPGA cluster resulted in a speed-up range from 15.5 to 20.6 [11]. The conclusion was that using multiple FPGAs can help relieve the I/O communication overhead which resulted in higher performance [11]. This information has provided the group with an estimate on how much of a speedup can be expected from the kth nearest software algorithm with a FPGA cluster. To show a definite speed up, however, we must know the performance of the current design from previous semester groups.

Another way of parallelizing algorithms or applications is with the use of GPUs. Researchers programmed the *k*-nearest neighbors algorithm with OpenCL on GPUs and in Verilog on a FPGA [2]. Both of these systems were seen to improve speed of the execution [8]. The algorithm on a FPGA has shown a decrease in energy consumption while the use of GPUs with this algorithm has shown the opposite despite achieving the same speedup [8]. The GPU implementation is also slower because multiple kernels are executed sequentially but must share a global memory [2]. The only downside of the FPGA implementation that was mentioned in this paper is the amount of time to create these complex systems with a hardware description language [2].

A similar design to what we will be attempting to develop has been created by other researchers: to have a cluster of FPGAs representing parallel cell units that utilizes a main controller board for high-level control tasks and accessing main memory [12]. Ratsaby and Zavielov [13] have provided an implementation that has shown the need for a hardware-software combination. With the data compressed to a realizable size, the arithmetic computations and nearest neighbor estimates can be done easily on a single FPGA chip [13]. The speed-up is significant compared to a software-only implementation. These represent the general idea to which our group will work towards in an attempt to meet the project specifications.

## Concept

The majority of the software developed for this project will be created, simulated, and programmed with Xilinx Vivado IDE and Xilinx SDK. Additional contributions include continued pipeline work that is being done by Martin Cowley as well as the guidance of Dr. Lina Sawalha. Below is the block diagram of our design. It shows general flow of logic through the cluster. The lines provide a visualization for directional flow of communication from device to device.

## Figure 1: Block Diagram of the FPGA Cluster Hardware Accelerator. The master Zybo Board will communicate to the slave Nexys 4 FPGA boards via Ethernet.

## Alternative Approaches

The design approach discussed will most likely not be the final design. There are many alternative approaches to creating an FPGA cluster to solve the kth nearest algorithm. Having more or less slave FPGAs could result in a better speedup or make designing the cluster simpler. Deciding how many CPU cores are going to be used, what the cores will be doing, and how the processor will complete the task needed are all different approaches for designing the software in the cluster.

An alternative approach to speeding up the kth nearest algorithm would be to use GPUs rather than FPGAs. However, this is not the focus of the project and the materials needed for this approach are not accessible. Studies have also shown that FPGAs can perform similar speedup with less power consumption[8].

## Precedence Matrix

**Figure 2: Precedence Matrix. Insures the completion of the hardware accelerator.**

## The precedence matrix shown in figure 2 helps organize the different components of the entire system. By highlighting the dependencies among these components, the ideal sequence to complete each section of the project is determined.

## Hardware Components

Below are the components that are to be used to complete the design as specified. These will be used in tandem with the software components to create one cohesive cluster. The hardware is to be optimized so that the software and hardware together handle data in the most efficient way possible.

|  |  |  |
| --- | --- | --- |
| **Hardware Components** | **Quantity** | **Description** |
| Nexys 4 | 2 | Artix-7 FPGA, 10/100 Ethernet PYE |
| Zybo Z7 | 1 | Artix-7 FPGA, 10/100/1000 Ethernet PHY, 1GB DDR3 Onboard Memory |
| Ethernet Switch | 1 | 100MB capability, At least 4 ports |
| Ethernet Cable | 3 | At least 1ft, Cat6 |
| Power Cable | 4 | AC to DC power supplies (5VDC at 2.5A) |
| Plastic Case | 1 | Encapsulates Hardware Components |

**Table 1: Hardware To Be Used**

## 

## Software Components

Below are the software components that will be implemented on some of the hardware listed above, most namely the Nexys 4 and Zybo Z7. As described above, in order to have a FPGA cluster that is as efficient as possible, the hardware and software need to work together in a cohesive manner.

|  |  |  |
| --- | --- | --- |
| **Software Components** | **Location** | **Description** |
| Data Sorting Algorithm | Zybo Z7-10 | Bubble Sort Algorithm Implemented In C |
| Euclidean Distance Calculator | Zybo Z7-10, Nexys 4s | Vector Distance Finding Algorithm Implemented in Verilog |
| Data Streaming Unit | Zybo Z7-10 | Controls Data Flow Between DRAM, Processors, and FPGAs. Implemented In C. |
| Execution Time Unit | Zybo Z7-10 | Measures The Time Between The Start And Finish Of The Kth Nearest Neighbor Algorithm. Implemented In C. |
| Xilinx Power Estimator | Main Computer | Determine power and cooling specifications to avoid overdesign or under-design of the cluster |
| Xilinx Vivado SDK - ZYNQ | Main Computer | Software Development Kit |

**Table 2: Software To Be Used**

## Models and Simulations and Design Validation

Simulations that we will perform for this project will focus mostly around the timing simulations using Xilinx Vivado. This will provide timing diagrams along with other data that will provide the speed-up of the new design over the previous design. We will also use Xilinx Power Estimator (XPE) for determining the power necessary for an efficient design.

## Testing Procedures

Some things, like the power cables and ethernet cables, need to be tested first to ensure proper functionality. From there as progress continues on the project, more hardware and software elements will be tested for functionality as well as efficiency. Below shows the components that are to be tested to ensure the FPGA cluster functions as specified.

|  |  |
| --- | --- |
| **Component** | **Testing Procedure** |
| Nexys 4s | Implement Simple Verilog Code and Confirm Correct Results |
| Zybo Z7 | Run Cowley and Irwin’s Hardware Accelerator and Confirm Correct Results |
| Ethernet Switch & Ethernet Cables | Test with a known working laptop and network |
| Power Cables | Test with a known working device |
| Data Sorting Algorithm | Input a test data set with known results and see if the results of our algorithm are correct |
| Euclidean Distance Calculator | Input a test data set with known results and see if the results of our algorithm are correct |
| Data Streaming Unit | Input a test data set to the processor and see if the FPGA receives the correct values |
| Execution Time Unit | Use this implementation on Cowley and Irwin’s accelerator |

**Table 3: Hardware Testing Table. This lists the components that will need to be tested throughout the project to determine if the design is functioning as specified.**

# Plan of Work

## Work Flow Diagram



**Figure 3: Workflow Diagram. The order of steps that must be completed in order to complete the hardware accelerator.**

## 

The workflow diagram above is a visual showing the work progess that has been completed to date. For a more representative visual of what the flow of events will look like for the remainder of the project, our Critical Path Network (CPN) should be referenced. This can be seen on the following page. It provides specific events along with the time it will take to complete each individual task.

## Critical Path Network



**Figure 4: Critical Path Network (CPN). Provides a time table for next semester’s events to be completed.**

The durations and times shown in the above CPN exclude most weekends between the start and end date shown. There are 110 days to complete the project in the given times, 80 of which are weekdays.

The earliest event time for each event is found by adding a preceding event’s early time with the duration to arrive at the selected event. This is done for all preceding events, and the largest value is taken. The starting event has event time 0.

E.g. event G. Its predecessors have early times of 2 and 1. Both have a duration to G of 4.5. The early time for G is 6.5 (2 + 4.5) because it is the larger sum of a preceding time and duration.

The latest event time for each event is found by subtracting the duration to arrive at a following event from that event’s late time. This is done for all following events, and the smallest value is taken. The final event has the same early and late event time.

E.g. event B. The following events have late times of 11 and 5 with durations 4.5 and 2, respectively. The late time for B is 3 (5 - 2) because it is the smaller difference of a following time and duration.

The independent float for each event is its latest minus earliest event time. An example of this is shown in the CPN Key.

## CPN Key

|  |
| --- |
| 1. Starting point 2. Installation of Xilinx Vivado and SDK is complete 3. Acquired hardware accelerator code from previous design project 4. Acquired ZYBO and Nexys 4 boards 5. Acquired ethernet switch 6. Acquired ethernet cables and power supplies 7. Made any necessary adjustments to baseline code 8. Verified operation of baseline code 9. Created / acquired dataset of training and test vectors 10. Added logic / instructions to measure system performance 11. Benchmarked baseline hardware accelerator on physical board 12. Found MAC addresses of each board in cluster 13. Tested data transfer between boards via ethernet switch 14. Created program to test data transfer across cluster 15. Added logic for bus communication signals 16. Added instructions to manage multiple FPGA blocks 17. Verified data transfer between boards 18. Updated instructions to efficiently stream data to each FPGA 19. Changed logic parameters depending on streaming data size and order 20. Tested system to verify data streaming from memory to FPGAs 21. Tested system to verify distance calculations in FPGAs are returned to the processor 22. Tested system to verify all distances are accounted for and the k nearest are selected 23. Updated FPGA logic to pipelined design (developed by Martin Cowley) 24. Final verification of system accuracy is complete 25. Benchmarked FPGA cluster performance 26. Compiled results and conclusions 27. Captured pictures of system operation 28. Rough draft of final project report is complete 29. Creation of graphs and drawings is complete 30. Proofreading of report is complete 31. Adjustments to final project report are complete and report delivered |

# Project Deliverables

Deliverables for this project should include results of tests run to determine the speedup and power consumption of the group’s optimized design, as well a working FPGA cluster. The boards will be held in one or more cases and the test results will be presented in report and presentation format with figures showing timing diagrams in addition to well commented verilog (.v) and (.c) files.

# Conclusions

As this project is a continuation of previous semester senior design students, most of the hardware programming has already been completed. The pipelined architecture is still being worked on, so the work to be done by our group is mainly focused towards the optimization of the current architecture and design. While there are only a few main areas of focus for this optimization, there are many different things that can be done for each. The two methods to be used for optimization are the parallelism of the instructions in hardware and the type of communication BUS. Parallelization of hardware will allow more instructions to be executed in a shorter amount of time. This is important as the more data points the machine learning algorithm has to work with, the more precise the results. The decision of where to store the data and how it is fetched will contribute to the speed of the processes. This step is hindered by the bitrate of a communication BUS. Serial communication will not be as efficient as data can only be transmitted one direction so a parallel BUS, which will allow data to flow in opposite directions simultaneously, is the best option.

# References

[1]“GPU vs FPGA Performance Comparison”, Berten Digital Signal Processing, 2016

[2]F. Muslim, A. Demian, L. Ma, L. Lavagno and A. Qamar, "Energy-efficient FPGA Implementation of the k-Nearest Neighbors Algorithm Using OpenCL", *Position Papers of the 2016 Federated Conference on Computer Science and Information Systems*, 2016.

[3]J. Bresnick, "Top 4 Machine Learning Use Cases for Healthcare Providers", *HealthITAnalytics*, 2017. [Online]. Available: https://healthitanalytics.com/news/top-4-machine-learning-use-cases-for-healthcare-providers. [Accessed: 10- Oct- 2017].

[4]S. Kilts, *Advanced FPGA design*. Hoboken, NJ: Wiley, 2007, pp. 1-16.

[5]L. Kim, S. Asaad and R. Linsker, "A Fully Pipelined FPGA Architecture of a Factored Restricted Boltzmann Machine Artificial Neural Network", *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 1, pp. 1-23, 2014.

[6]M. Kurek, T. Becker and W. Luk, "Parametric Optimization of Reconfigurable Designs Using Machine Learning", *Lecture Notes in Computer Science*, pp. 134-145, 2013.

[7]L. Kim, S. Asaad and R. Linsker, "A Fully Pipelined FPGA Architecture of a Factored Restricted Boltzmann Machine Artificial Neural Network", *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 1, pp. 1-23, 2014.

[8]Y. Liu, J. Bi and Z. Fan, "Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms", *Expert Systems with Applications*, vol. 80, pp. 323-339, 2017.

[9]E. Bortnikov, A. Libov and O. Shacham, "WMU – Central Authentication Service", *Libproxy.library.wmich.edu*, 2017. [Online]. Available: http://libproxy.library.wmich.edu/login?url=https://search.proquest.com/docview/1904493074?accountid=15099. [Accessed: 23- Oct- 2017].

[10]Z. Zhang, "Introduction to machine learning: k-nearest neighbors", *Annals of Translational Medicine*, vol. 4, no. 11, pp. 218-218, 2016.

[11]D. Kim, J. Jung, T. Nguyen, D. Kim, M. Kim, K. Kwon and J. Jeon, "An FPGA-based Parallel Hardware Architecture for Real-time Eye Detection", *JSTS:Journal of Semiconductor Technology and Science*, vol. 12, no. 2, pp. 150-161, 2012.

[12]A. Ahmadi, H. Mattausch and T. Koide, "A parallel hardware design for snake models with an fpga architecture", *NSIP 2005. Abstracts. IEEE-Eurasip Nonlinear Signal and Image Processing, 2005.*, p. 13, 2005.

[13]J. Ratsaby and D. Zavielov, "An FPGA-based pattern classifier using data compression", *2010 IEEE 26-th Convention of Electrical and Electronics Engineers in Israel*, 2010.