Submitted on September 8

Enabling Real-Time Roadside Visual Object Detection Using FPGAs

by

Team Visionary

Jinkyu Lee

Xinqi Wang

Dongyeon Woo

Advised by

Dr. Peter Milder

Table of Contents

Introduction

1.1 Executive Summary

According to US Vehicle Registration statistics, the number of total registered vehicles has been increasing from 255.9 million in 2008 [3] and up to 287.3 million in 2020 [8]. As more vehicles get on the road and faster transportation systems become imminent, highway management and road safety constantly pose new challenges. For instance, the Federal Highway Administration of the U.S. Department of Transportation reports that there are approximately two million wildlife-vehicle collisions annually in the United States. On high speed roads, not only animals, even approaching pedestrians can be hard to notice and avoid when driving at a high speed.

Over the past years, computer vision has been an emerging field in machine learning, and object detection has been explored and utilized in many applications. While most object detection methods utilize deep neural networks for their performance efficiency, real-time object detection applications, which detect objects in streaming video and are expected to generate immediate outputs, have been struggling to achieve a desirable speed while also preserving adequate accuracy. The solution proposed in this project is to develop a system that implements one of the state-of-the-art real-time object detection algorithms to detect potentially hazardous road objects and alert drivers to take immediate action when necessary.

1.2 Project Statement and Testbed for Evaluation

One of the main objectives of this project is to develop a more efficient and economical object detection system using application-specific hardware such as field programmable gate array (FPGA) since heavy deep neural network computations often result in bottlenecked performance and high power demands. Once the software prototype of the system has been tested on appropriate datasets, preferably customized according to the expected functionality of the detection model, a hardware accelerator will be designed and implemented on the FPGA development board to improve the performance. Aside from being able to operate independently of a CPU driver, the high specialization of FPGAs to even bit-level customization makes them especially suitable as DNN accelerators.

The object detection model we chose to work with is YOLO(You Only Look Once), which is one of the state-of-the-art real-time object detection models. It has been explored and developed

frequently since its introduction in academica. Because of its on-stage characteristics, YOLO is able to detect objects in an image at a significantly faster rate than many two-stage object detection algorithms, such as the RCNN family, while still achieving a competent accuracy. Since its development, several variations of the YOLO model have been proposed by the authors of YOLO. While some improvements necessarily increased the complexity of the neural network model, other variations aimed to implement simpler models to reduce computation demand while still accurate enough for real world applications. Since the original YOLO model often fails in detecting small objects, which is a desired feature for our system, newer versions, including YOLOv3, SlimYOLOv3 and YOLOv3-Tiny, will be considered instead and the trade-offs of each evaluated to choose an appropriate model that best meets the requirements of this project.

The expected output of this project is going to be a real-time object detection system, implemented using FPGA, that is able to efficiently capture image data and detect objects in the image to determine potential hazards on the road. The power consumption will be estimated using evaluation tools in Vivado and available libraries will be used to record and measure the system's speed performance. Finally, the FPGA implementation will be compared to that of CPU or GPU to evaluate the efficiency of the final system.

1.3 Objectives & Goals

- ➢ The project will implement a deep neural network-based object detection algorithm (e.g., YOLOv3, SlimYOLO, etc.) determined according to the desired systems features and application setting.
- ➢ The system will utilize an FPGA (combined into a single development board) to maximize speed and improve energy efficiency.
- ➢ The system will be able to communicate results to the outside world in order to make use of the output data.
- ➢ This project will evaluate the speed of the final system and its expected power cost. The computational performance, including frame rate and overall latency, and the expected power needed to complete the same amount of tasks, will be compared to that of general purpose systems, typically run on CPU or GPU.

1.4 Background

Software

A convolutional neural network (CNN) is a common type of neural network. It takes an input image data and passes through its layers. The hidden layers between the input and output layers are typically convolutional layers that abstract input images to a feature map. The convolution or cross-correlation operation reduces the number of free parameters, allowing the network to be deeper with fewer parameters. This model has been widely used for the Computer Vision applications as a popular way of handling object detection. In order to efficiently implement the detection process with a limited amount of resources, the detection process based on a deep learning model should be less computationally heavy because it is directly related to lowering power consumption and latency. To keep high accuracy for real-time applications, You Only Look Once (YOLO) CNN model was developed in 2015 [6]. This model requires only a single forward propagation through the neural layers to detect objects on input images but still performs better than the other well-known detection models, therefore, it will be a great software model for our project.

Hardware

FPGAs are semiconductor devices that are made of configurable logic blocks (CLBs) connected via programmable interconnects. Therefore, FPGA can deliver a reconfigurable architecture with its programmability even during runtime, allowing the user to reconfigure the data types and data path easily for large degree of parallel computations to measure up the CNN's high computational demands across each layer. Furthermore, FPGA's any-to-any I/O connection ability enables itself to connect to any peripheral devices without a host central processing unit (CPU), giving it more adaptability for different applications [1]. Traditionally, neural network models have been implemented on graphic processing units (GPU) since it was proven to be more efficient than CPU in 2004 [4]. However, the limitation of GPU comes from the fact that its data flow is defined by software and directed by the GPU's memory hierarchy, which causes itself to have additional latency and power consumption for memory access, lowering performance/watt rate [1]. Therefore, in terms of cost, power, performance and reconfigurability, using FPGA is a better choice for the hardware aspect of this project . However, the remaining challenge is that designing logic blocks to implement digital systems on FPGAs is different from programming and compiling on CPUs or GPUs. In other words, using FPGA requires various computer-aided engineering software tools (CAE tools) to

automate the designing, debugging, simulating, synthesizing and *place-and-routing digital systems on FPGA. The CAE tools chosen for this project will be described further in the following Tools section.

**Place-and-Route is to interconnect logic elements on the grid of the FPGA.*

1.5 Social, Environmental and Societal Impact considerations

Once our design is fully implemented, it will be placed on roadsides and used to alert drivers at high speed. By deploying our work, it will minimize collisions between vehicles and animals and directly help to protect the ecosystem and make our driving experience much safer, which will also help reduce our auto-insurance costs.

Our work will also help to reduce the cost of maintaining the highway as such the number of roadkill decreases. As our deployment proves to be effectively preventing potential roadkill, the concept of wildlife-vehicle collision will hopefully be left as history.

Proposed Solution

2.1 Narrative

Pre-Trained YOLOv3 Object Detection Model

The chosen open-source pre-trained model is the standard YOLOv3 for object detection based on Darknet [7]: (https://github.com/pjreddie/darknet/tree/master/src). Our current goal for this project is to increase the efficiency using an FPGA based hardware accelerator. Therefore, retraining this YOLOv3 model with a fully customized dataset made of local wildlife images will be one of the stretched goals of our project after accomplishing to make a hardware accelerator for the YOLOv3 model. One of our goals to achieve regarding this popular CNN model for this project is to study the details and algorithms at arithmetic level behind their specific method for detecting multiple objects in real-time. Therefore, this pre-trained model will be thoroughly utilized for this project to improve object detection by merging it with an FPGA based hardware accelerator. Therefore, this YOLOv3 model, which was updated for better accuracy than that of the original YOLO, will be a great object detection model for this project.

Xilinx Vivado Software Tool

Hardware design and implementation will be done using XilinxVivado. One of the main goals of this project is to develop a FPGA accelerator for the real-time object detection system implemented based on the YOLOv3 model by analyzing the computation and memory requirements of the software prototype and DNN parameters. Since non-computing processing might require additional memory and resources and insufficient management of on-chip memory will cause frequent off-chip data transfer, the utilization and design of the FPGA resources must be considered carefully in order to yield desirable acceleration results while also maintaining expected accuracy. Design modules will be written in Verilog or SystemVerilog and synthesized using the Vivado synthesis tools to generate netlists to be fitted on the targeted FPGA board. Simulation and verification of the final hardware design will be done with Xilinx SDK, an IDE for development of embedded software applications targeted towards Xilinx processors. Vivado also provides tools to estimate total system power consumption and summary of on-board resource utilization to help analyze design characteristics and further optimization.

MicroZed

The MicroZed™ FPGA development board shown on the right will be used for implementing YOLOv3. This board has a Xilinx Zynq®-7000, which has FPGA programmable logics paired with 32-bit Advanced RISC Machine (ARM) Cortex-A9 MPCore [9]. Along with Vitis AI Development Kit mentioned earlier, Xilinx is the biggest distributor of FPGA, and ARM is the most widely used instruction set architecture. Therefore, MicroZed FPGA board with its popular hardware specifications will easily lead not only this project but also the big-scale implementation of this project in the future at economical scale. Furthermore, it has an integrated camera connection peripheral, which will shorten the required time for implementing a camera and verifying the functionality of the YOLOv3 model for this project. The total estimated power consumption is 4.8W according to the documentation[2]. The exact power consumption rate will also be measured by observing how different it is for our specific case with the pre-trained YOLOv3. Therefore, in consideration of its popular SoCs and development environment, on-board camera connection peripheral and low-power consumption rate, MicroZed will be a great hardware candidate for this project.

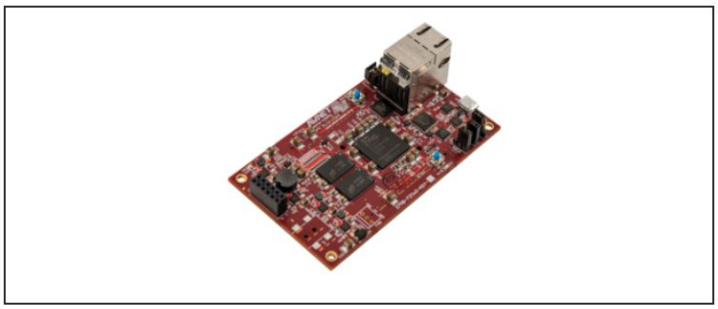

Figure 0. This picture shows a MicroZed FPGA Board

2.2 Project tasks and road-map to deliverables

a) Software and Hardware components for the full working system

i. Fully Understanding YOLOv3 and Darknet

ii. Software prototype

- Build customized framework for DNN computations
- Choose appropriate dataset to verify the functionality of the system (PASCAL VOC, COCO, VisDrone, etc)
- *Stretch Goal : customized dataset to accommodate the environment targeted in this project.*
- Implement software prototype (C / C++ / Python)
- Verify/Optimize software prototype functionality

iii. Hardware accelerator

- Design hardware implementation for the DNN model (Inference task)
- Implement a trained model
- Simulate and verify correct functionality of the hardware design
- Test on Microzed
- Optimization (e.g., Off-chip memory management/movement)

b) A thorough evaluation of the system

i. Verification of the final system

- Evaluation of the system's performance and cost characteristics
- Compare characteristics (e.g., power consumption, computational performance) of the system with general purpose hardware (e.g., CPU or GPU)

c) A short video demonstrating the system's functionality
d) Full documentation of how to load, configure, and test the system

2.3 Potential Risk and Mitigation Strategy

The main objective of this project is to design and implement a hardware accelerator, using FPGA, to improve the system's overall performance and power consumption. However, because the DNN model considered in this project is relatively large, additional efforts must be invested in investigating current accelerator structures in the research field to develop a desirable design that best accommodates the model structure in this project. We will also try to be flexible in model choices as long as the potential performance and accuracy of the system is not impaired. Also, since hardware compilation and debugging often requires more time and effort and the potentials of FPGA acceleration of YOLOv3 have not been explored thoroughly, the hardware development stage of the project is expected to consume large amounts of time and efforts and thus will be guided in frequent consultation with the advisor to make sure the team is on the right track.

Due to the ongoing pandemic, we're currently using an online chat to communicate with each other and overcome the time difference. However, later when developing and implementing hardware designs, it will be difficult to make progress and collaborate if we cannot work directly in person and labs. Thus, effective and frequent communication and updates will be necessary to keep everyone involved and on the same page. We will also try to divide certain tasks carefully to make sure it can be done by the person in charge if collaboration is difficult to achieve (e.g., testing FPGA board). Plans will be modified accordingly as the project progresses.

Management Plan

3.1 Project team and Primary Responsibilities

Xinqi Wang <*is responsible for designing the hardware system***>**

- Has experience with embedded systems and VHDL from ESE380 and 382.
- Worked on a deep learning hardware accelerator project where a reinforcement learning application was designed and the software prototype was accelerated by implementing the DNN model using FPGA.
- C/C++, Verilog/VHDL

Dong Yeon Woo <*is responsible for verifying the functionality of the software prototype and simulating and debugging the hardware system***>**

- Has experience of embedded systems in ESE280 and 381.
- C/C++, Assembly
- Interested in designing logic blocks and neural network models with FPGA.

Jinkyu Lee <*is responsible for implementing and optimizing the software prototype and assembling MicroZed and setting up petalinux on it***>**

- worked with embedded systems in ESE 280 and ESE 381.
- worked with VHDL in ESE 382.
- worked with an object detection model to implement it on FPGAs.
- likes programming and thrilled by getting correct results after debugging.

3.2 Project timeline

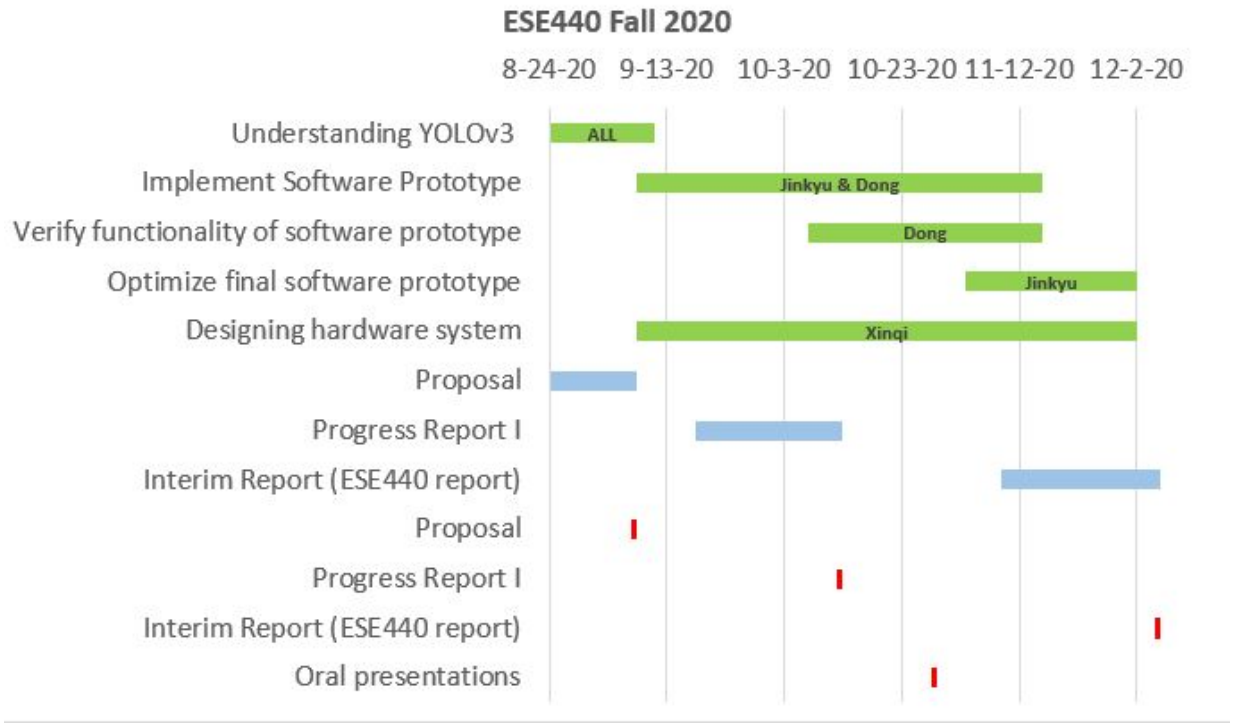

Figure 1. This chart shows our timeline and who is responsible for each task.

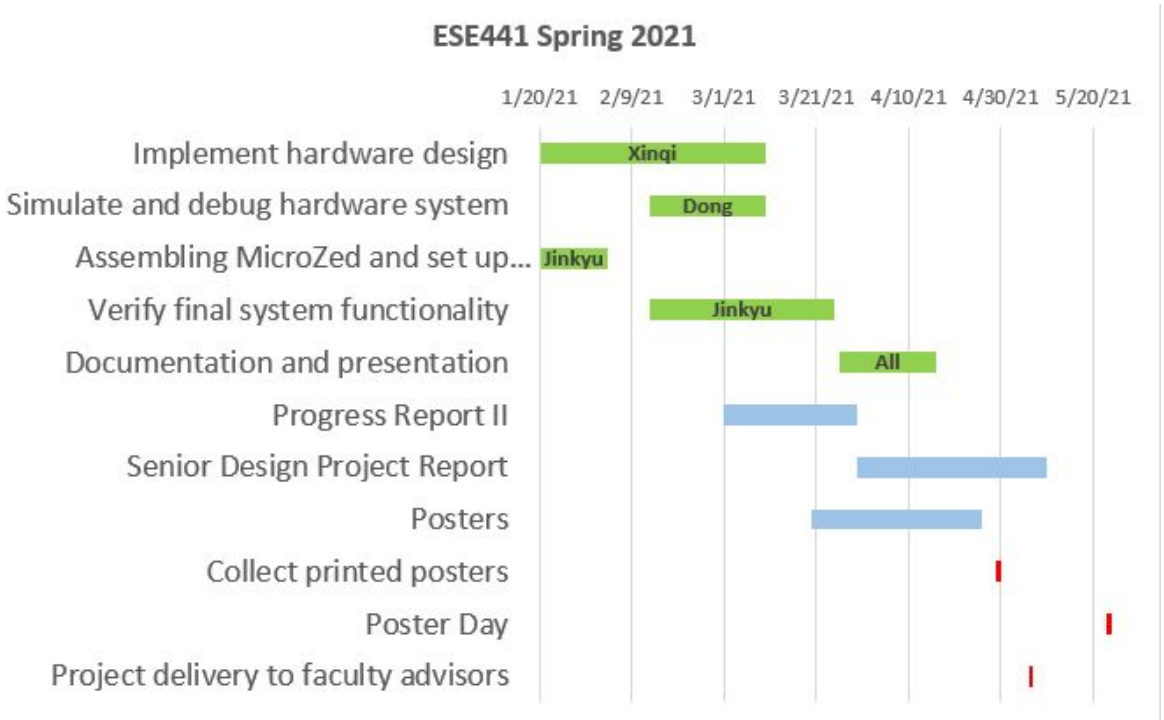

Figure 2. This chart shows our timeline and who is responsible for each task in Spring 2021.

3.3 Cost Summary (Budget)

Initial Implementation Summary Table

Payment Name	Cost (USD)	Count	Duration	Sub Total (USD)
MicroZed Embedded Vision Carrier Card Kit	306.01	1	-	-
MicroZed 7020 Embeddable System-on-Module with Standard ARM Cortex-A9 Processor	228.26	1	-	-
Camera Module Peripherals for MicroZed	499.00	1	-	-
Total				To be monitored **For now, we borrowed the parts from Professor Milder's Lab**.

Table 1: This shows the exact amount for each cost and the sum.

References (Bibliography)

[1] F. Fallahlalehzari, “FPGA vs GPU for Machine Learning Applications: Which one is better? - Blog - Company,” Aldec. [Online]. Available: https://www.aldec.com/en/company/blog/167--fpgas-vs-gpus-for-machine-learning-applications-which-one-is-better. [Accessed: 28-Feb-2020].

[2] “MicroZed™ Zynq® Evaluation Kit and System on Module Hardware User Guide,” Zedboard, 2017. [Online]. Available: http://zedboard.org/sites/default/files/documentations/5276-MicroZed-HW-UG-v1-7-V1.pdf. [Accessed: 11-Mar-2020].

[3] “Number of U.S. Aircraft, Vehicles, Vessels, and Other Conveyances,” Number of U.S. Aircraft, Vehicles, Vessels, and Other Conveyances | Bureau of Transportation Statistics. [Online]. Available: https://www.bts.gov/content/number-us-aircraft-vehicles-vessels-and-other-conveyances. [Accessed: 1-Feb-2020].

[4] K. Oh and K. Jung, “GPU implementation of neural networks,” ELSEVIER, vol. 37, no. 6, pp. 1311–1314, Jun. 2004.

[5] “Public Road Length, Miles by Ownership,” Public Road Length, Miles by Ownership | Bureau of Transportation Statistics. [Online]. Available: https://www.bts.gov/content/public-road-length-miles-ownership. [Accessed: 1-Feb-2020].

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[7] J. Redmon, YOLO: Real-Time Object Detection. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 12-Feb-2020].

[8] “US VIO Vehicle Registration Statistics, Fast Quote on Car Data,” Hedges & Company, 06-Dec-2019. [Online]. Available: https://hedgescompany.com/automotive-market-research-statistics/auto-mailing-lists-and-marketing/. [Accessed: 1-Feb- 2020].

[9] “Zynq-7000 SoC Data Sheet: Overview,” XILINX, 02-Jul-2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf. [Accessed: 2-Mar-2020].

Appendix A (Meeting Schedule)

Team Visionary will have a weekly meeting with Professor Milder at 10AM on Wednesday.