ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

Infernet: An FPGA Inference Accelerator Service

1.0 Introduction

We would like to build a distributed neural net (NN) inference service using a combination of hardware and software. This service could be thought of as the machine learning FPGA IP offerings available from Xilinx and Intel running on Amazon's F1 cloud FPGA service [1, 2, 3].

The basic components of the project - cloud networked FPGAs and ML inference IP - already exist, as mentioned previously. Due to the scope of our project and the limited time available to us we do not expect to compete with those existing technologies. Instead, the goal of our project is to combine them in an interesting distributed system that can leverage cloud-networked hardware for ML tasks.

2.0 Project Team

2.1 Andrew Uderian

My background as a computer engineer is split between software and hardware. I worked as an embedded C/C++ intern at a startup called Nanoleaf, and did my PEY at Intel FPGA (formerly Altera). There, I worked on the External Memory Interface IP available within Quartus - my projects included a silicon test screen for exercising hardened logic and porting a legacy IP generation flow to the latest versions of Quartus. With my experience I bring development and debug skills in both embedded software and FPGA IP.

2.2 James Liu

I was a part of UofT's Blue Sky Solar Racing design team for 3 years, and served as Firmware Lead and Electrical Lead across two working solar cars. I bring to this project all my experience architecting low level real time firmware, working with RTOS, and making devices talk to each other. I also did my PEY at Rambus where I completed numerous design flow automation projects outside of my regular analog IC layout work, so I am familiar with the structure and pitfalls of large EDA software tooling.

2.3 Quinn Smith

What I bring to the team is a solid background in FPGA tooling/compilers as well as cloud systems. I did research with Professor Chow a few years ago and then went on to do my PEY at

Intel FPGA. I worked on Quartus with a team that specialized in improving software performance and building customer features such as partial reconfiguration. After working at Intel I did work at AWS working on a service which offered databases in the cloud (Amazon RDS).

2.4 Team Summary

All of our team members have solid and varied backgrounds in hardware, software, and FPGA tooling. In previous projects and roles, we have each written testbenches, unit tests, and integration tests. However, we collectively lack and will need to develop knowledge around working with AXI protocols - our previous experiences have mostly used Intel FPGAs and the Avalon protocol.

Finally, we all have exposure to ML via various undergraduate-level intro courses. Combined with our technical abilities, we have sufficient skills to execute the project as proposed.

3.0 Project Description:

The system would be composed of the following components networked together:

- One or more inference accelerators (IA)
- A load balancer (LB)
- A desktop client (client)
- A desktop GUI, which will be implemented within the client to display results

A user in control of the desktop client will be able to execute an ML inference task using the following steps:

1) Using the desktop client, the user sends a request to the LB.
2) Upon receipt, the LB queries the on-network IAs to find one which is currently not busy.
3) If an IA is available, the LB responds to the client with the chosen accelerator's IP.
4) The desktop client streams data to the IA.
    a) The system is designed to allow for batch jobs, to minimize the impact of the query/allocation overhead.
5) The IA sends the results of the inference job to the desktop client.
6) (For display purposes) A GUI on the desktop displays the inference results.

Two system diagrams are presented below, as Figures 3.1 and 3.2. The former shows the planned system at a high-level, including the workflow described above, and the latter shows a more detailed view of the system.

The ML task that the system will target is classification of handwritten digits, using the MNIST dataset [4]. The specific task being solved and the accuracy of the IA will depend on the IP core itself, and could be changed by using a different NN core. This project targets an accuracy rate of 80% on MNIST, considering the size limits of the available FPGAs.

To validate that the system correctly implements the designed functions and features, the workflow described above will be used. More specifically, the test will comprise the desktop client contacting the LB, being allocated an FPGA, streaming multiple images from the MNIST dataset to the allocated device, and receiving classifications back with an accuracy of at least 90% - comparable to initial historic attempts at solving the task [4].

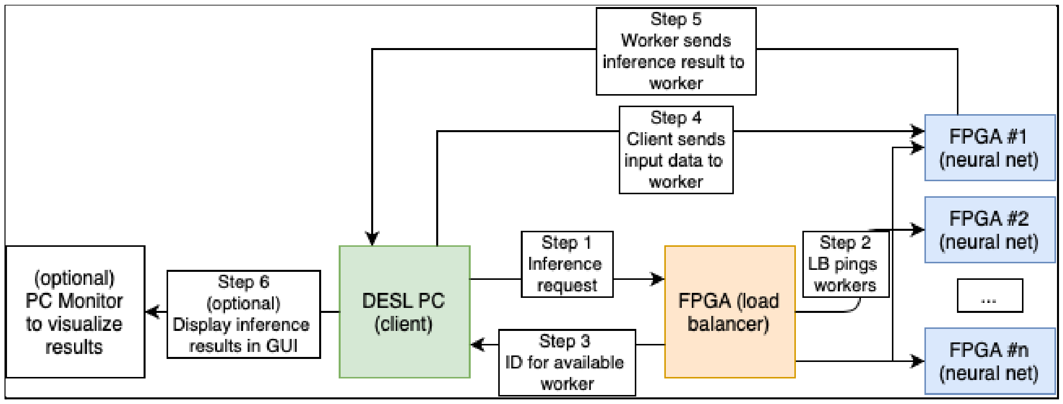
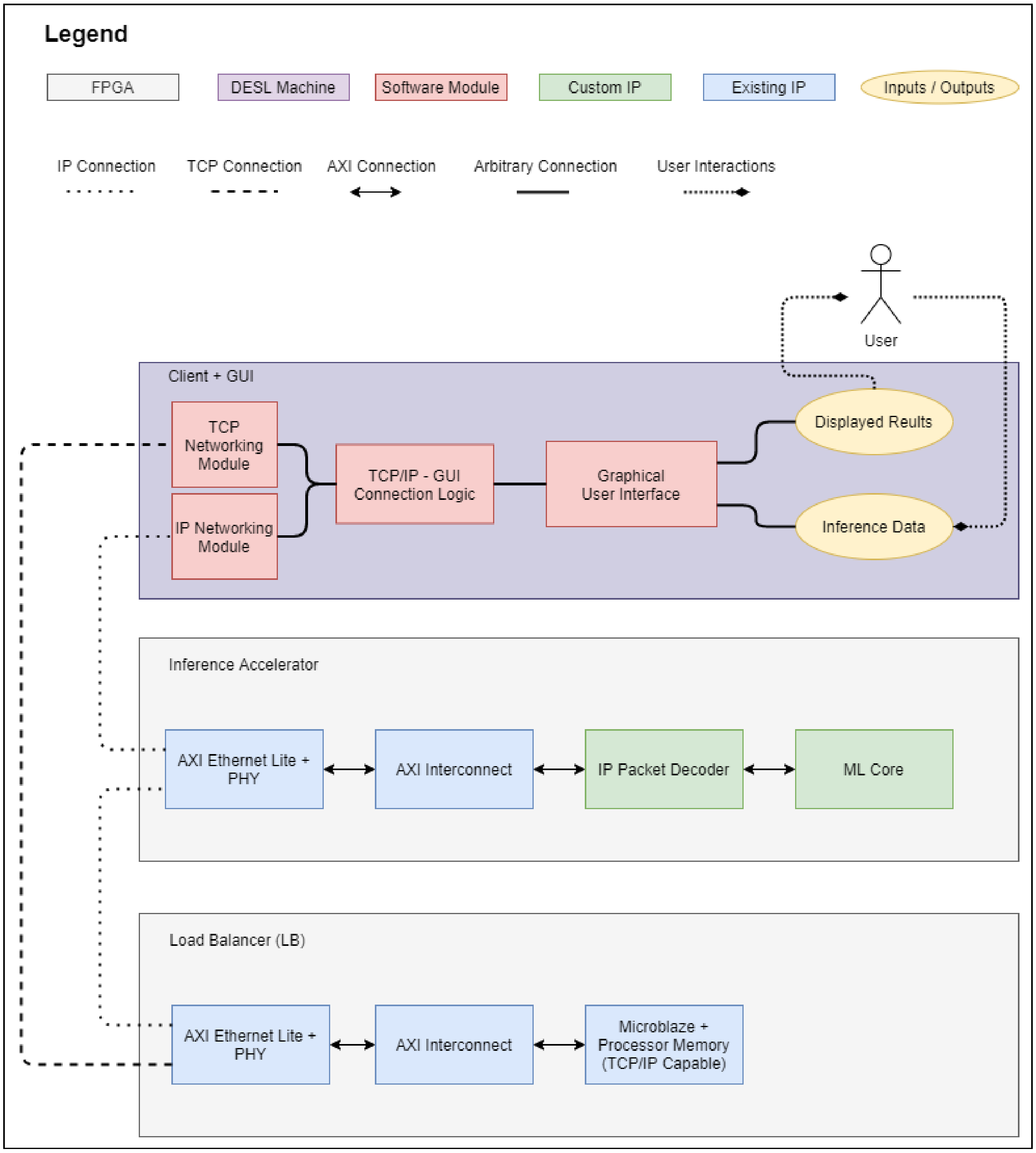


Figure 3.1 System Diagram with Steps

ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

Figure 3.2: Detailed System Diagram



ECE532: Project Proposal

Team 6
Andrew Uderian
James Liu
Quinn Smith

ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

4.0 Testing

4.1 Inference Accelerator

The functionality of the IAs will be validated using multiple testbenches. During development, their individual components - the IP packet decoder and ML core - will be individually validated using mock components. More specifically, simple “black-box” models of each component will be developed and used to validate them individually.

Once individually validated, the custom IP cores will be integrated and simulated together to verify they work together. Finally, a simple desktop test Python client will be developed to simulate both the full-featured final client and load balancer for hardware testing. The goal will be to use the mock client to verify the accelerator can receive, process, and send IP workloads correctly.

4.2 Load Balancer

The load balancer will be primarily based on the basic echo server developed for the initial group demo. There are no anticipated changes to the base hardware of the server to build the LB - i.e., its development is expected to be an embedded software endeavour. The changes to the basic software provided for the echo server project will be exercised with test cases during early validation. Once development is complete, a simple desktop test Python client (possibly the same client described for validation of the IAs, as described in 4.1 above) will be developed to simulate the full-featured final client and IAs for hardware testing. The goal will be to verify that the LB can receive, process (by querying the IAs), and respond to requests correctly.

4.3 Client and GUI

The client and GUI will be a single program that can be run on the DESL machines. Because the client will target the x86 desktop environment, software unit tests will be straightforward to develop and will be used to validate the two components. Once complete, we anticipate the other components of the system - i.e., the LB and IAs - will be complete and validated, as shown in the proposed milestone schedule in section 8.0. As such, we will perform final testing of the desktop client using the real LB and IAs.

4.4 Overall System

The test plans/methodologies outlined in 4.1, 4.2, and 4.3 are sufficient to exercise the system in its entirety.

ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

5.0 Project Complexity:

The project, as proposed, meets the suggested complexity score for a team of 3. A detailed estimate of the complexity score is presented in Table 5.0 below.

Table 5.0: Project Complexity Estimation

Complexity Category / Component	**Points**
Desktop to FPGA network connection Bidirectional communication using raw IP packets between desktop client/LB (request) and desktop client/IA (workload).	0.25
FPGA Network Connectivity Bidirectional communication using raw IP packets between the desktop client and LB via a MicroBlaze (inference request) on the LB. Bidirectional communication using raw IP packets between the desktop client and IAs without a MicroBlaze.	0.25 1.00
IP Core Implemented In Hardware The IP packet decoder and ML inference cores are anticipated to be of moderate complexity.	~1.00
Software in MicroBlaze The LB will be implemented using embedded C on a MicroBlaze.	0.25
Meaningful visualization of program run/statistics gathered/results obtained Inference results and aggregate statistics will be displayed in the desktop Python client's GUI.	0.75
Total	**3.5**

6.0 Risks:

The greatest risk to the project is that the team does not have the time to engineer and test fault tolerance in the system, especially in the IP packet core. Within the scope of the project this risk is appropriate, given that the network will be small and should only contain traffic from other boards and PCs on the network - which are controlled by other teams. If the IP packet core fails to be sufficiently tolerant for the given conditions, the team will use a MicroBlaze running LWIP within the IA design as a fallback.

The team has also scheduled one week of buffer/slack to mitigate unexpected bugs, integration issues, or implementation complexity.

If the project proceeds strongly, the team is considering developing a 4th system component: a desktop daemon for the DESL machines to program their attached FPGAs with the IA's bitstream on-the-fly. This would allow the system (controlled by the LB) to dynamically expand during operation, given un-allocated boards. This module is planned to be optional - the team will allocate time to it once the other components are working as expected.

7.0 Resource Requirements:

As of the submission of this document we have no other resource requirements. If we succeed in implementing the daemon described in section 6.0, we anticipate possibly needing special permissions on the DESL machines for it to run.

ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

8.0: Milestones

Milestone #	**Andrew**	**James**	**Quinn**	**Group**
1	Do assignment #1: write an AXI core	Do assignment #1: write an AXI core	Do assignment #1: write an AXI core	Plan milestones
2	Simulate a basic XOR NN core Simulate architectural components of the full NN core	See if the MicroBlaze can handle multiple TCP connections without an RTOS If needed, get an RTOS working on the MicroBlaze	Simulate the IP packet core Convert AXI-stream chunks into data packets to send to a mock NN core	
3	MNIST classification (NN core) passing simulation Develop data format for IP packet module communication with NN core (with Quinn) Integrate with IP packet module (with Quinn) Develop “busy” functionality for ML core - ML core can signal to its IP packet module whether it has completed the current data frame	Develop data format for LB communication with IP packet module (with Quinn) Integration testing with IP packet module (with Quinn)	Develop data format for IP packet module communication with NN core (with Andrew) Integration with NN core (with Andrew) Develop data format for LB communication with IP packet module (with James) Integration testing with LB (with James) Modify IP packet module to handle NN core being “busy”	Do Assignment #2 (due 0.5 weeks after Milestone 3) DEMO Goals: Mock desktop client to send 1 frame to an IA (without LB involvement) and get inference result back Mock desktop client to send machine request to LB and get a (spoofed) machine # back
4	Develop daemon to program DESL boards via network	IP packet implementation for full-fledged client	Basic GUI module for client (might swap with	

	commands from LB		James)	
5	-	-	-	System integration!
6	slack	slack	slack	

ECE532: Project Proposal
Team 6
Andrew Uderian
James Liu
Quinn Smith

9.0 Bibliography

[1] **"Amazon EC2 F1 Instances"**, Amazon Web Services, Inc., 2021. [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/. [Accessed: 30- Jan- 2021].

[2] **"AI Inference Acceleration"**, Xilinx, 2021. [Online]. Available: https://www.xilinx.com/applications/megatrends/machine-learning.html. [Accessed: 30- Jan- 2021].

[3] **"Flexibility: FPGAs and CAD in Deep Learning Acceleration - Intel"**, Intel.com, 2021. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01283-flexibility-fpgas-and-cad-in-deep-learning-acceleration.pdf. [Accessed: 30- Jan- 2021].

[4] Y. LeCun, C. Cortes, and C. J. C. Burges, **"THE MNIST DATABASE of handwritten digits"**, *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 31-Jan-2021].