ECE532 Project

Team 6: Infernet

Meet the team

Andrew

James

Quinn

What is Infernet?

- Distributed network of machine learning inference modules targeting the MNIST dataset
- Multiple networked FPGAs with hardware inferencers available to desktop PC clients
  - For this project: FPGAs and PCs on FPGANet

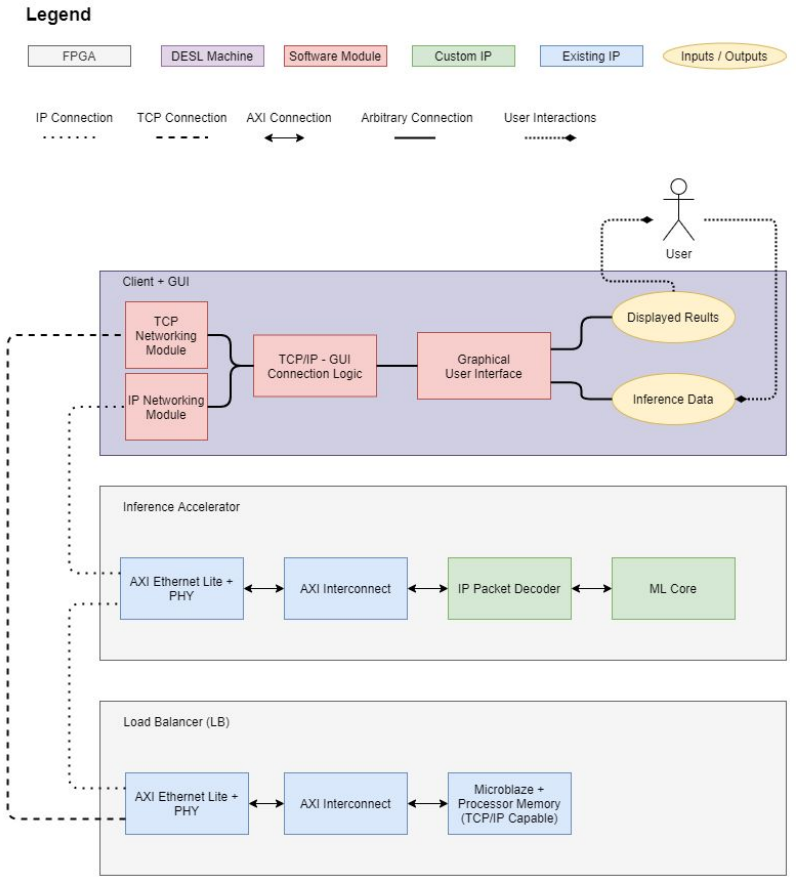
Sample images from the MNIST dataset

Why make Infernet?

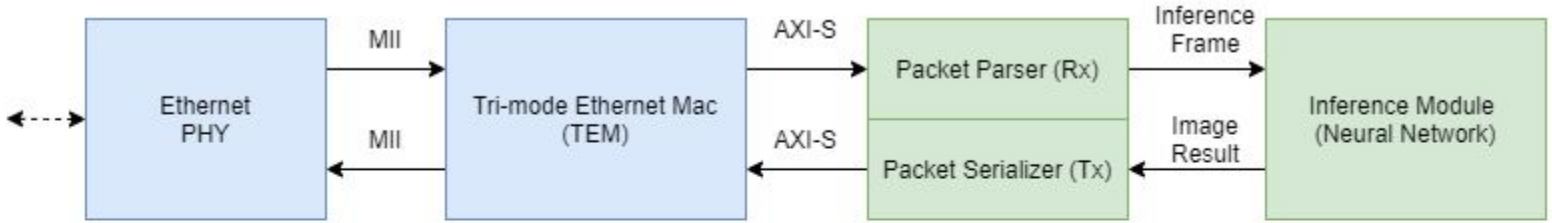
- Proof of concept: in theory could work on bigger networks if we had bigger FPGAs
- Mimics common real-world “cloud”/”compute-as-a-service” models
  - e.g. Amazon F1 instances - FPGA shells
- Potential applications in edge compute where local resources can’t provide sufficient inference throughput

System Overview

- 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



Inference Accelerator



Legend

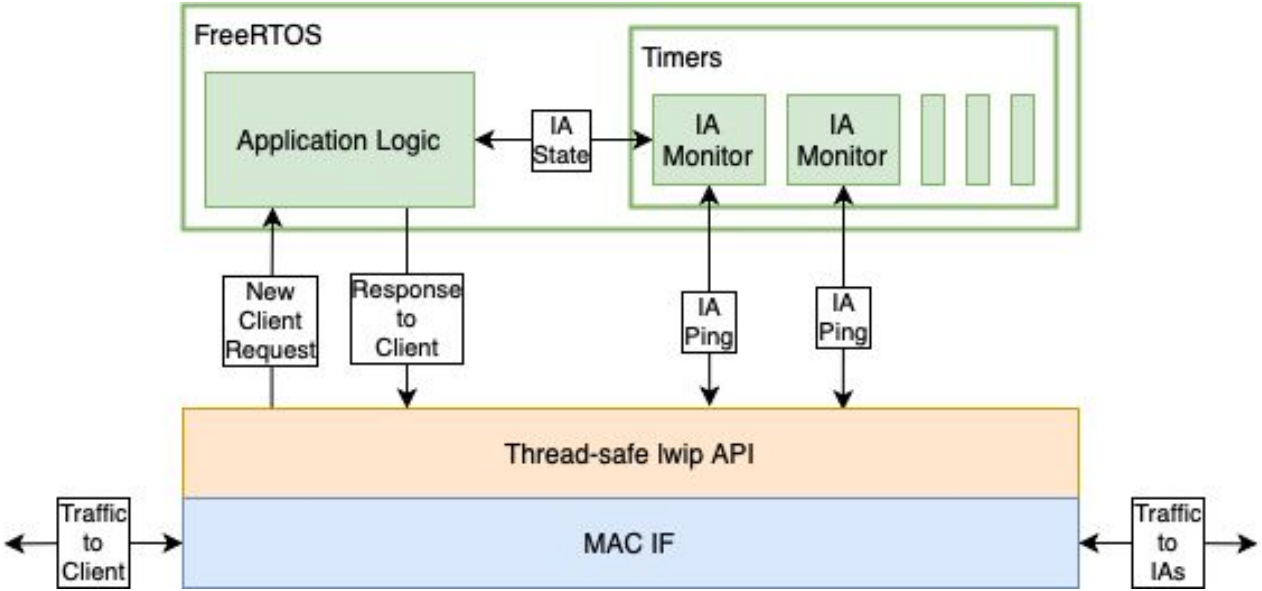
Custom IP

Existing IP

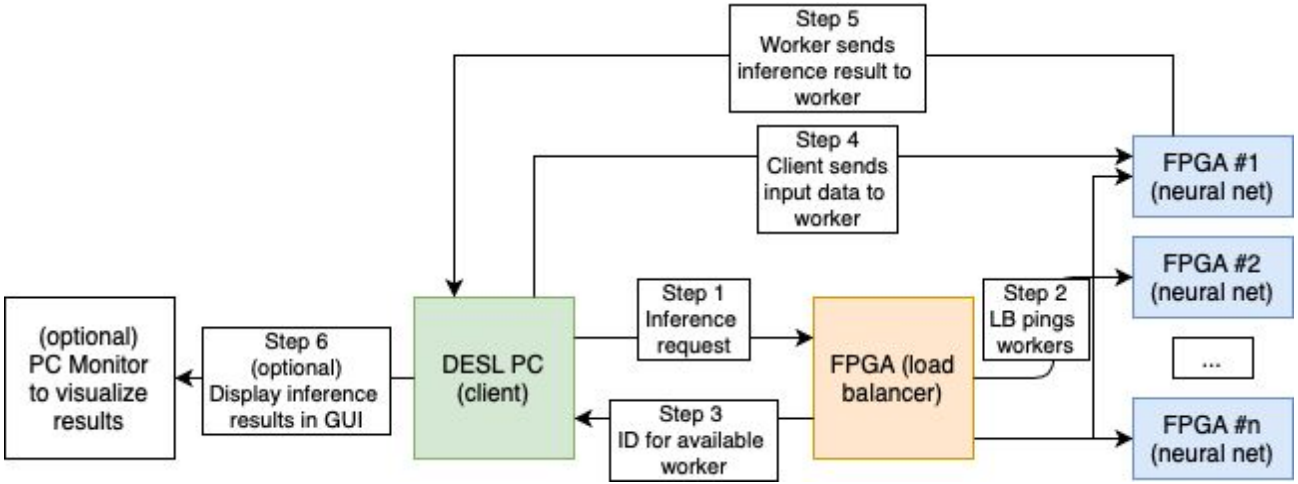
Module Connection

Network Connection

Load Balancer



User Workflow



Milestones

Milestone #	Andrew	James	Quinn
1	Assignment 1 + Proposal	Assignment 1 + Proposal	Assignment 1 + Proposal
2	Simulate neuron microarchitecture. Train potential NN candidates in SW.	Discovery work: can RTOS work on microblaze?	Get IP Packet Tx + Rx Simulated
3	MNIST Core Passing Simulation Integrate with Rx + Tx	Python module for sending and receiving inference data packets	Integrate Tx + Rx with TEM Help Andrew with Tx + Rx
4	Develop Daemon (Optional) otherwise help with debug / client	Networking logic for client, Load balancer firmware	GUI logic for client
5	System Validation	System Validation	System Validation
6	slack	slack	slack

Test Plan - IP Packet Tx Rx

1. Module level simulation (primitives ,rx/tx, rx/tx + TEM)
2. Validate ethernet + IP frames can be sent from SW->SW with minimal loss/degradation
3. Validate in hardware by sending/receiving message from dummy software module

Test Plan - Hardware Neural Net

- Python prototype acts as a “golden model”
  - Can easily extract data for test frames from specific neurons or layers to debug

1. Module-level simulation - fixed-point multiplier, convolution
   - We are here
2. Simulate convolution layers against golden model with sample MNIST frames
3. Validate accuracy of entire NN core against golden model
   - Use same test set as during training/model eval

Test Plan - Load Balancer

- ❏ Python simulators for client and accelerators
- ❏ Run a selection of test cases to verify correctness
- ❏ Run a randomized stress test for a length of time, while monitoring for resource leaks and correctness

Test Plan - System

- Integration testing of NN core with IP packet core in sim, then HW
- If inference accelerator can talk to both client and load balancer correctly then our system is functional.
  - Resolved in integration tests
- System level testing will focus on scale
  - Multi-client, multi-accelerator

Project Risks/Uncertainties

Inference Accelerator

- Impact of fixed-point arithmetic on inference accuracy
- Potential on-chip routing issues for inference module
  - Depends on NN design
- Ethernet frame fault tolerance

Load Balancer

- Lwip Socket API may not work in Xilinx SDK
  - Xilinx bugs :'(

Risk Mitigation

- Within the NN, can generally trade off resources for accuracy
  - Data width ⇔ accuracy
  - Number of DSPs (NN size) ⇔ accuracy
- Additional fault tolerance in Rx/Tx module (spicy UDP)
- In the worst case, LB could be implemented with the RAW lwip API and a bare metal superloop -- at the expense of one extra week of time

Open Questions

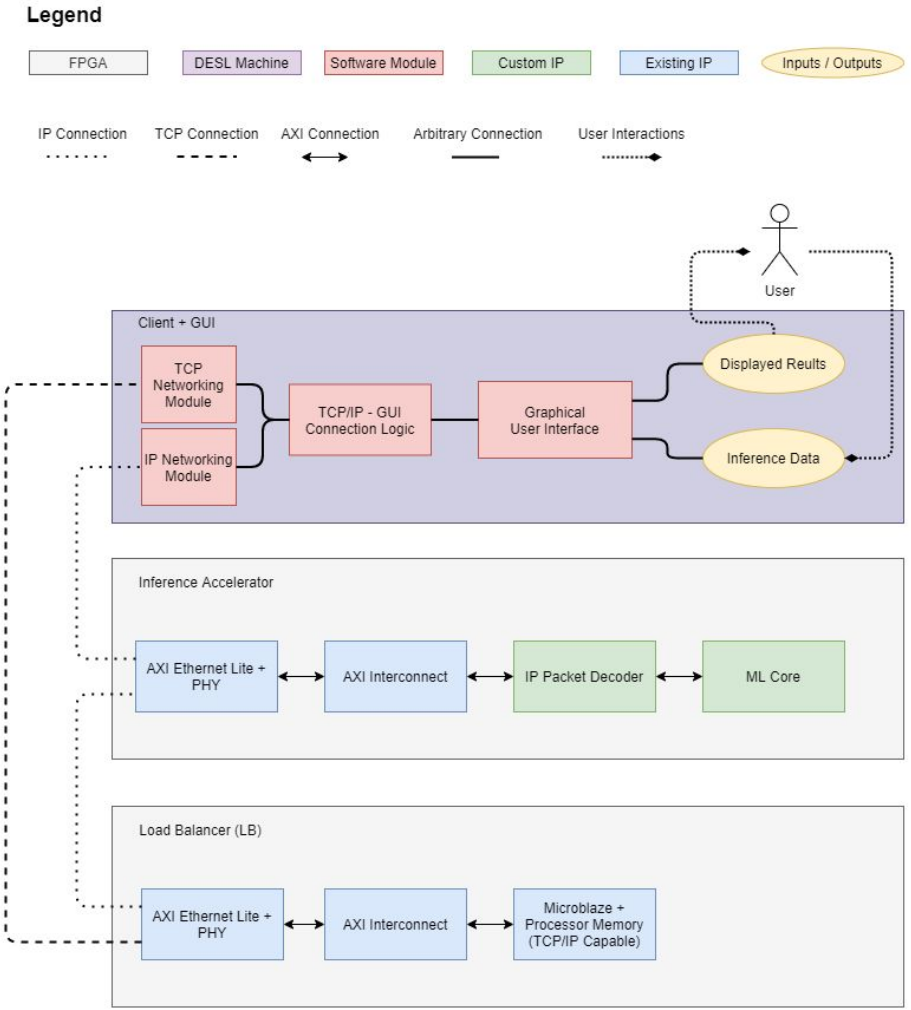
Inference Accelerator

- Final NN architecture not decided
  - Ongoing experiments in software!
  - Size/complexity ⇔ accuracy
- How do we configure the Tri-Mode Ethernet MAC to suit our needs? (RTM!)

Load Balancer

- It would be more scalable if LB could discover the accelerators on the network
- Could the LB handle all the work of sending the data?
  - Single point of contact for the client

Questions?


Legend
FPGA
DESL Machine
Software Module
Custom IP
Existing IP
Inputs / Outputs
IP Connection
TCP Connection
AXI Connection
Arbitrary Connection
User Interactions
User
Client + GUI
TCP Networking Module
IP Networking Module
TCP/IP - GUI Connection Logic
Graphical User Interface
Displayed Reults
Inference Data
Inference Accelerator
AXI Ethernet Lite + PHY
AXI Interconnect
IP Packet Decoder
ML Core
Load Balancer (LB)
AXI Ethernet Lite + PHY
AXI Interconnect
Microblaze + Processor Memory (TCP/IP Capable)