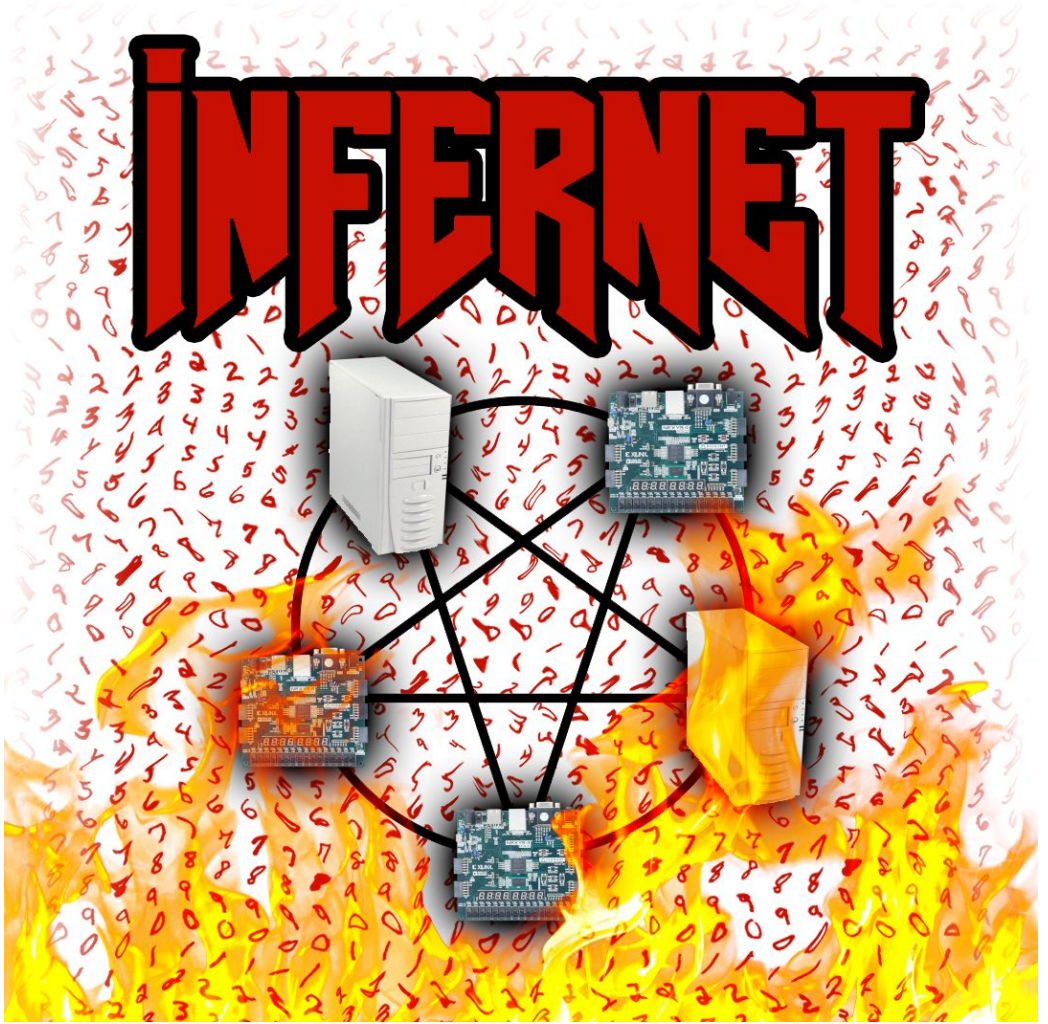
Team 6
ECE532 Project



Meet the team

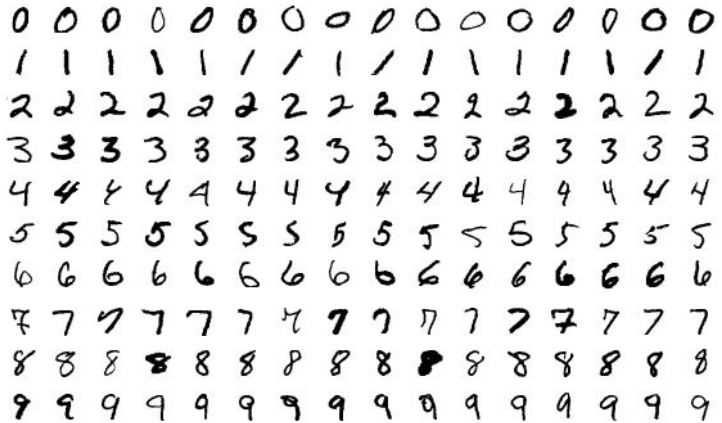
Andrew

James

Quinn

What is Infernet?

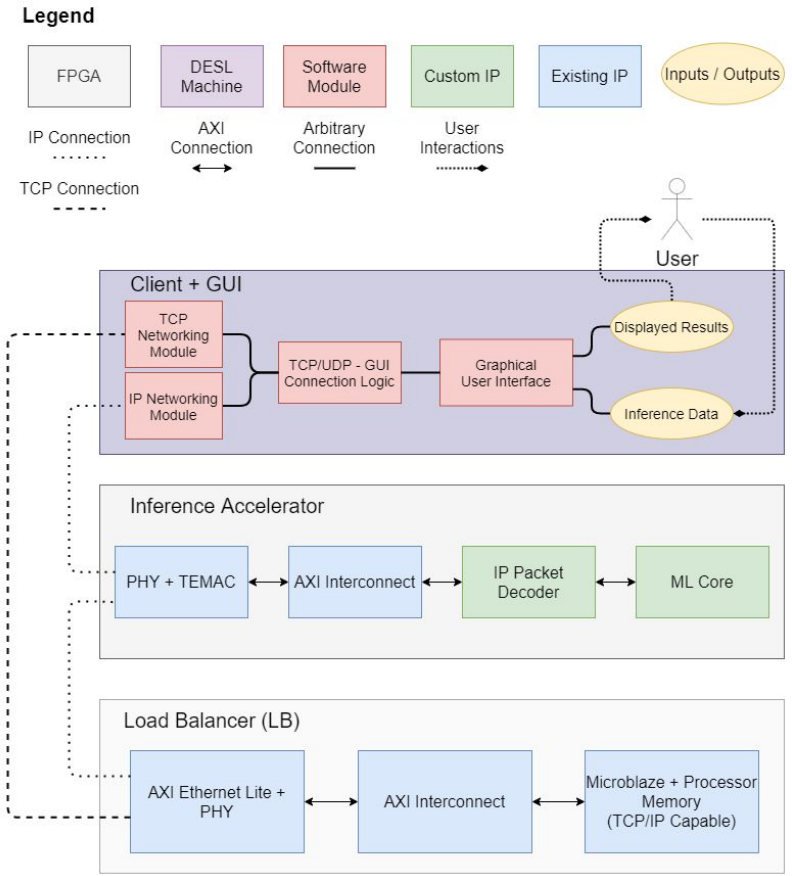
- Distributed network of machine learning inference modules targeting the MNIST dataset
- Multiple networked FPGAs with hardware inferencers available to desktop PC clients
  - For this project: FPGAs and PCs on FPGANet



Sample images from the MNIST dataset

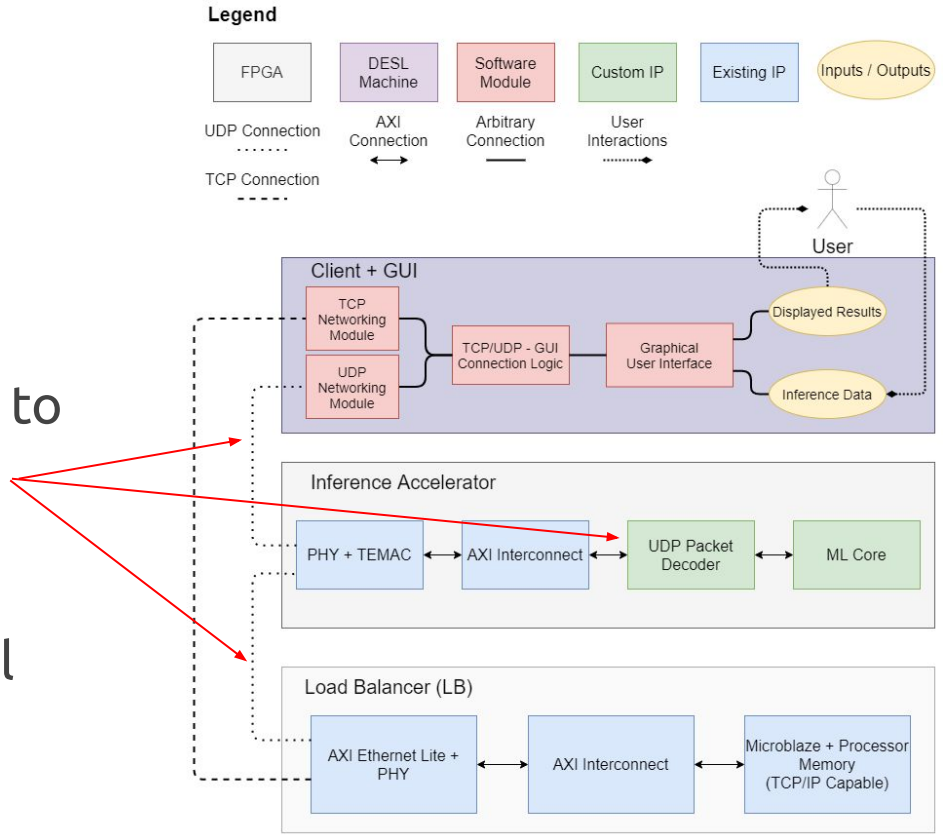
System Overview - Old

- 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

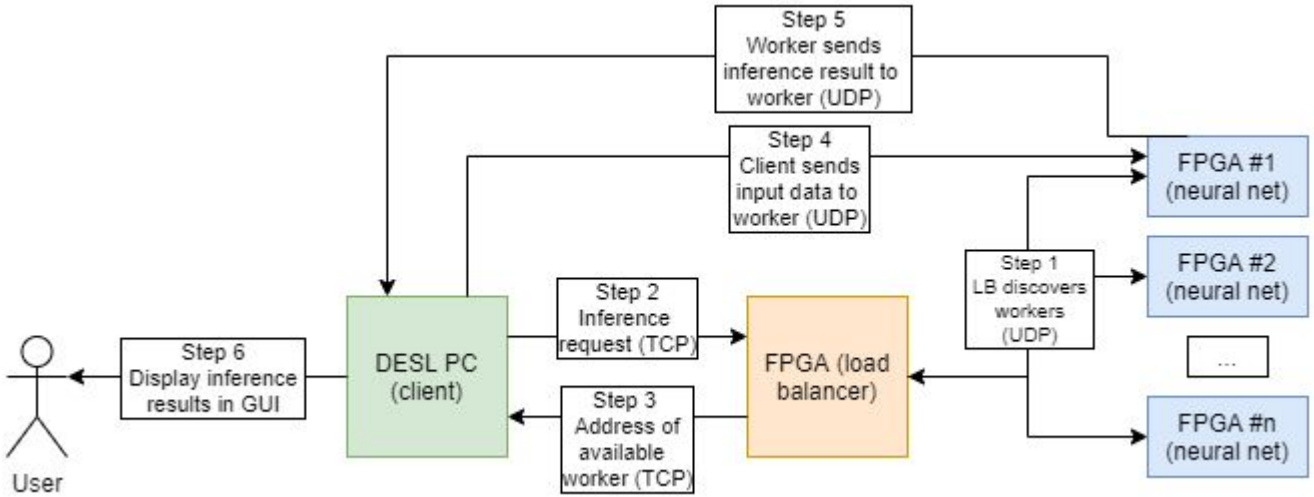


System Overview - New

- All the same system components
- Switching from IP to UDP to circumvent RTOS issues
- Same amount of communication, slightly different protocol



Updated Protocol



Challenges So Far

Client Software:

- Wrangling network interface names in windows
- No way to open a raw socket from python socket library because not admin

Load Balancer:

- Due to Vivado bug, FreeRTOS and lwip socket API do not work together.
- No raw IP in lwip

IP Packet Module

- Scary Integrations
- Network Order / Endianness
- Physical Interfaces

Hardware Neural Net

- Lots of stuff to build
- Understanding DSP slices/columns and carry chains

Future Challenges

Client Software:

- Creating a good-looking GUI in python

Load Balancer:

- Get UDP working with no-OS raw lwip.
- Catch and fix corner cases in state machine

IP Packet Module

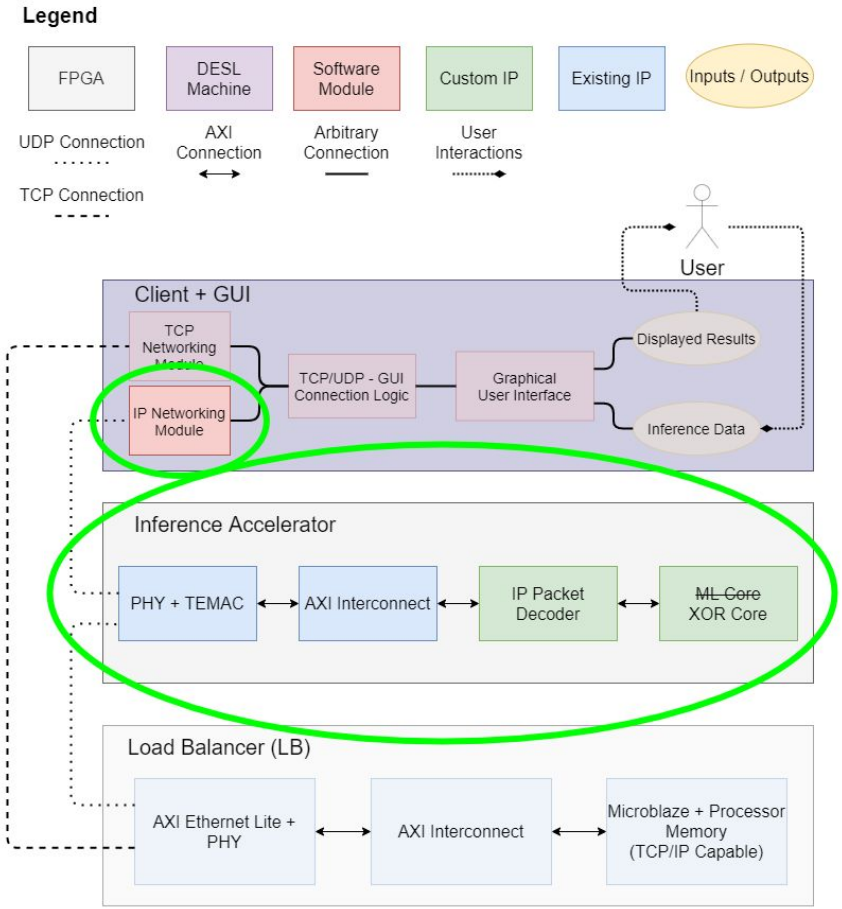
- Timing closure with big dataframes (workarounds exist)

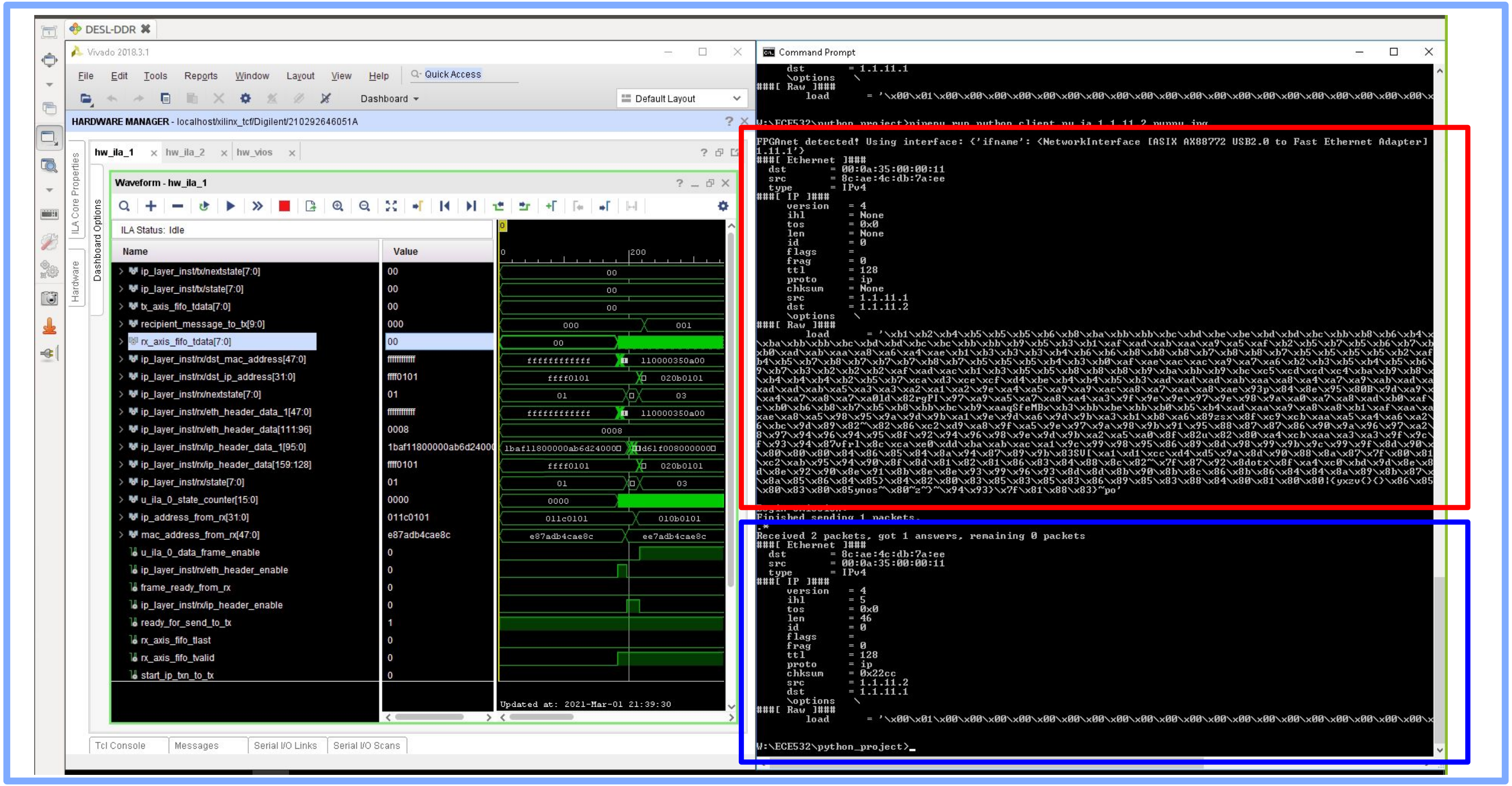
Hardware Neural Net

- Lots of stuff to build! :)
- Complicated control logic
- NN accuracy issues might be hard to debug
    - But also not fatal to the project

Our Demo

- Focused on getting FPGA <-> DESL PC networking working in hardware
  - Getting IP packet core from sim to H/W
  - Figuring out Python IP networking on DESL PCs
- Replaced full-scale neural net with XOR neural net
  - Uses same underlying dataflow arch (data width, mults)



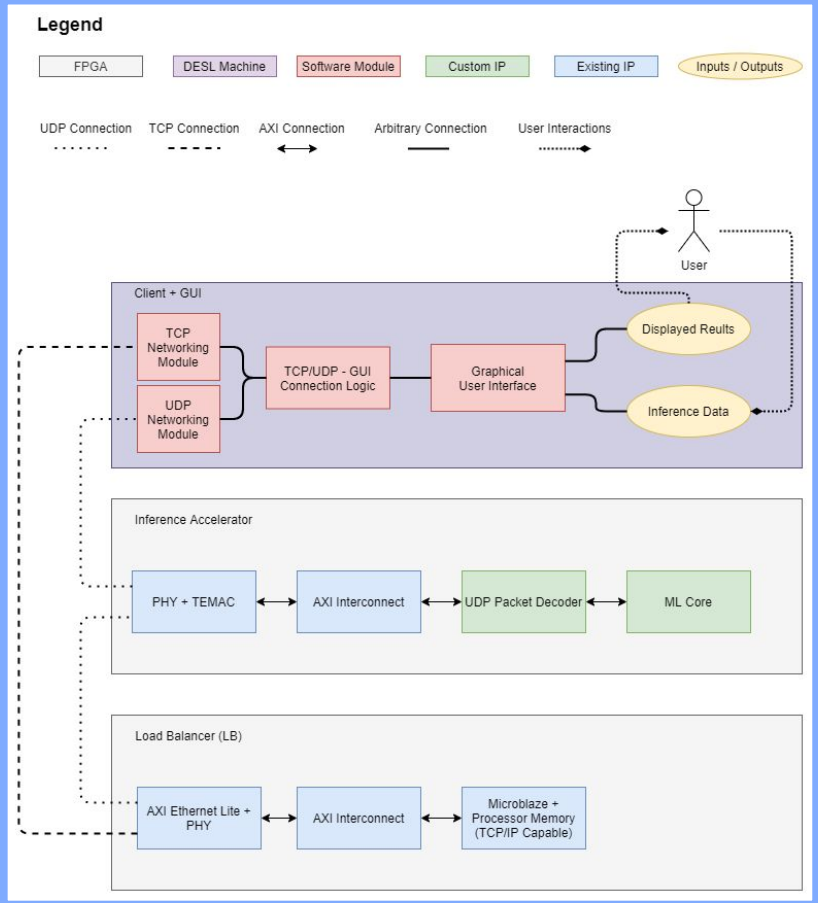

DESL-DDR
Vivado 2018.3.1
File Edit Tools Reports Window Layout View Help
Quick Access
Dashboard
Default Layout
HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210292646051A
hw_ila_1
hw_ila_2
hw_vios
ILA Core Properties
Dashboard Options
Hardware
Waveform - hw_ila_1
ILA Status: Idle
Name
Value
ip_layer_inst/tx/nextstate[7:0] 00
ip_layer_inst/tx/state[7:0] 00
tx_axis_fifo_tdata[7:0] 00
recipient_message_to_tx[9:0] 000
rx_axis_fifo_tdata[7:0] 00
ip_layer_inst/rx/dst_mac_address[47:0] ffffffffffff
ip_layer_inst/rx/dst_ip_address[31:0] ffff0101
ip_layer_inst/rx/nextstate[7:0] 01
ip_layer_inst/rx/eth_header_data_1[47:0] ffffffffffff
ip_layer_inst/rx/eth_header_data[111:96] 0008
ip_layer_inst/rx/ip_header_data_1[95:0] 1baf11800000ab6d24000
ip_layer_inst/rx/ip_header_data[159:128] ffff0101
ip_layer_inst/rx/state[7:0] 01
u_ila_0_state_counter[15:0] 0000
ip_address_from_rx[31:0] 011c0101
mac_address_from_rx[47:0] e87adb4cae8c
u_ila_0_data_frame_enable 0
ip_layer_inst/rx/eth_header_enable 0
frame_ready_from_rx 0
ip_layer_inst/rx/ip_header_enable 0
ready_for_send_to_tx 1
rx_axis_fifo_tlast 0
rx_axis_fifo_tvalid 0
start_ip_txn_to_tx 0
Updated at: 2021-Mar-01 21:39:30
Tcl Console
Messages
Serial I/O Links
Serial I/O Scans
Command Prompt
FPGAnet detected! Using interface: {'ifname': <NetworkInterface [ASIX AX88772 USB2.0 to Fast Ethernet Adapter] 1.11.1'}
###[Ethernet]###
dst = 00:0a:35:00:00:11
src = 8c:ae:4c:db:7a:ee
type = IPv4
###[IP]###
version = 4
ihl = None
tos = 0x0
len = None
id = 0
flags =
frag = 0
ttl = 128
proto = ip
chksum = None
src = 1.1.11.1
dst = 1.1.11.2
\options \
###[Raw]###
Finished sending 1 packets.
Received 2 packets, got 1 answers, remaining 0 packets
###[Ethernet]###
dst = 8c:ae:4c:db:7a:ee
src = 00:0a:35:00:00:11
type = IPv4
###[IP]###
version = 4
ihl = 5
tos = 0x0
len = 46
id = 0
flags =
frag = 0
ttl = 128
proto = ip
chksum = 0x22cc
src = 1.1.11.2
dst = 1.1.11.1
\options \
###[Raw]###
W:\ECE532\python_project>

What's Left

- ❏ Rebuild the LB using UDP, with the echo server as a template
- ❏ Convert IP core to UDP
- ❏ Networking validation
  - ❏ Validate LB<->IA protocol components
  - ❏ Validate LB<->Client protocol components
- ❏ Finish building the neural net in hardware
- ❏ Close timing on the neural net
- ❏ Write a pretty GUI

Final Demo Plans

- Multiple DESL PCs each contacting the load balancer, acquiring a board, and streaming large data batches to the accelerators
  - We'll see how far it can scale
    - Probably will end up limited by DESL infrastructure (e.g. how many PCs can we control at once?)
  - Hopefully demonstrating good classification accuracy for the ML workload
    - And hopefully no crashes :)


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

Questions?