Using CaST for FPGA design

RUTGERS
WINLAB | Wireless Information Network Laboratory

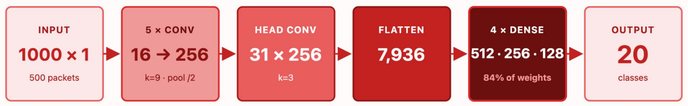
Charles Yu, Abraham Klein, Rishabh Ramaswamy, Dhruv Ramaswamy
Advisor: Richard Martin

Overview

Our goal in this project was to show CaST as a proof of concept and then run AI models on an FPGA, with WFNet.

Main accomplishments:
- Trained an MNIST Model on python and ran it using CaST to show CaST proof of concept
- Ran WFNET Model on FPGA to determine inferencing capabilities
- Developed WFNETv2 inspired by original WFNET improving accuracy and inferencing time
- Compare to CPU, GPU inference

Original WFnet Model (91% accuracy on testing data)

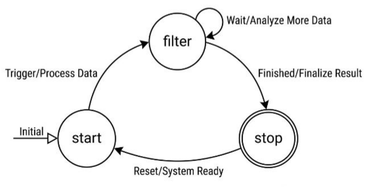


CaST

- CaST: (Channels and State): high-level alternative to System Verilog
  - Concise, better readability
  - Easier to learn
- CaST transpiler: Converts CaST code into System Verilog
- Anatomy: concurrent state machines connected by buffered channels
  - Interface block → defines channels
  - Shared block → defines shared variables between states
  - States block → defines state
  - machine.
  - Initial execution triggered by data channels
- Avoids clock-by-clock programming → easier to design
- Use of state machines and message passing → retains hardware control

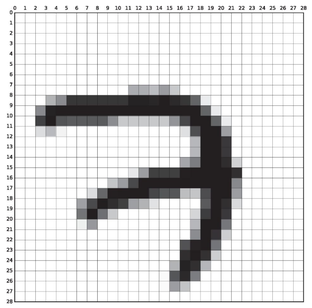
```
Machine
interface {
    input uint16 idx1;
    input uint16 mem_val;
    input commands trigger;
    output uint16 memory_idx;
    output uint16 memory_write_val;
}

shared {
    var uint16 num_reads;
    var uint16 val;
    var uint16 index;
}

states {
    read_data: index <- idx {
        ...
        goto filter;
    }
    filter: {
        ...
        goto read_data;
    }
}
```

Outline of CaST machine



Graph of CaST machine workflow

MNIST Model



(a) MNIST sample belonging to the digit '7'.

(b) 100 samples from the MNIST training set.

- Neural network that recognizes hand written digits (0-9)
- Input: 28 × 28 grayscale images
- Used to evaluate neural network inference performance on CaST
- Trained on python to get weights + biases
- Run on CaST
  - Uses hexagonal systolic matrix multiplication for increased efficiency
  - Uses a two layer perceptron structure

WFNETv2 Model

Model Purpose
Given packet data between a server and a client computer, determine the website

ubuntu

Collecting Data/Training methods

NumPy

- Scraped web data for 8 webistes tand gathered packet data using tcpdump, ethtool, xvfb, and other linux tools
- Used Playwright python API to click around websites to scroll, to send/recieve packets
- Cleaned /Normalized data using numpy, matplotlib
- Made 2 channel CNN using pytorch

Inferencing

- Ran toolchain to take model → put it on FPGA

1	2	3	4	5
PyTorch / Brevitas	hls4ml	Vitis HLS	Vivado / v++	Host app
Quantization-aware training of the MNIST CNN	Turns the exported ONNX graph into an HLS C++ project	Synthesizes that C++ into an RTL IP core with AXI ports	Place-and-route into a Zedboard bitstream or a U200 xclbin	Loads the design, streams images in, reads predictions out

Results

- Testing accuracy of 98% could be improved with convolution layers
- Cycles per image is constant, which is a benefit of FPGA
- Showed CaST can be used as a cleaner alternative to System Verilog

Results from MNIST CaST model

```
accuracy: 98<%
>cycles: 52981
cycles per image: 529.8
MACs: 440000 (8.3 per cycle)
```

MNIST Statistics on Alveo U200 FPGA

97.40%	4,916	203 µs
accuracy on 1,000 test images	inferences per second	per image

- For U200, we were only able to get MNIST model deployed, not WFNET or WFNETv2
- WFNETv2 has 83% accuracy in any 1024 packet window, 99% in any full load of a web page

Future Work

- Integrate CaST into more complex models like WFNET
- Add more features to CaST transpiler
- Train WFNETv2 on QUIC data
- Expand training breadth WFNETv2, adding more websites to classify
- Test WFnet on U200 and benchmark it

Acknowledgements

We would like to thank our advisor, Dr. Richard Martin, for his support and guidance. We would also like to thank WINLAB for the opportunity to work on this project.

WINLAB