Material Sorter

Final Demo

Presenters:

Sebastian Gomes
Callum McKelvie
Dhruv Sirohi
Zakria Nabi

Project Overview

Key idea:

- Camera takes photo of items on conveyor belt
- Algorithm makes classification decision using image from camera
- Servos move to push object into correct bin

Application:

- Originally focused on ML to classify materials for recycling
- After descoping, we developed an algorithm to sort objects based on colour
- Use-case: Detecting errors in product coloring on assembly line

Why FPGA?

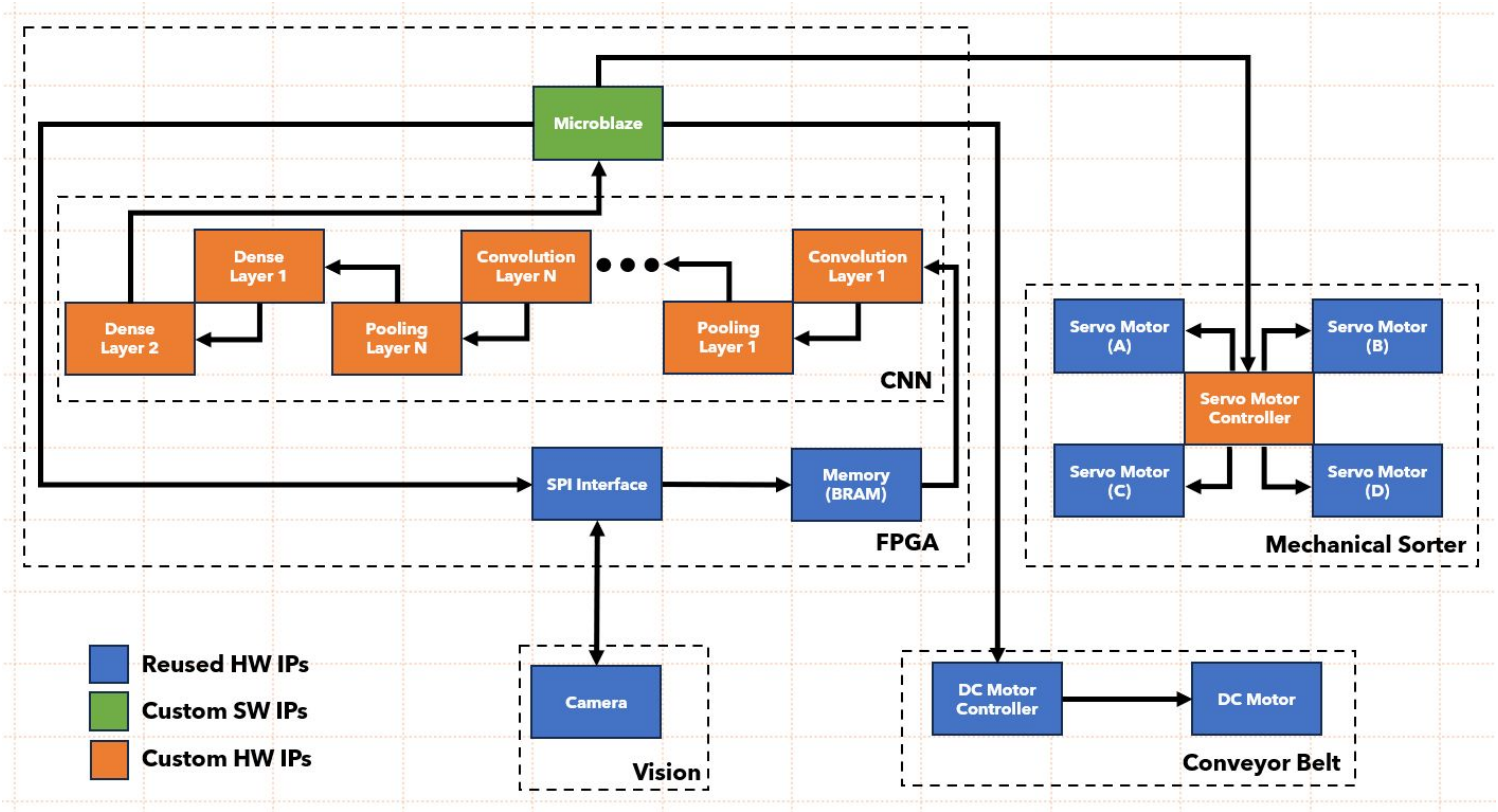
Machine learning algorithm for decision-making:

- Run compute-intensive inference tasks efficiently
- No need to run model on cloud or with GPU

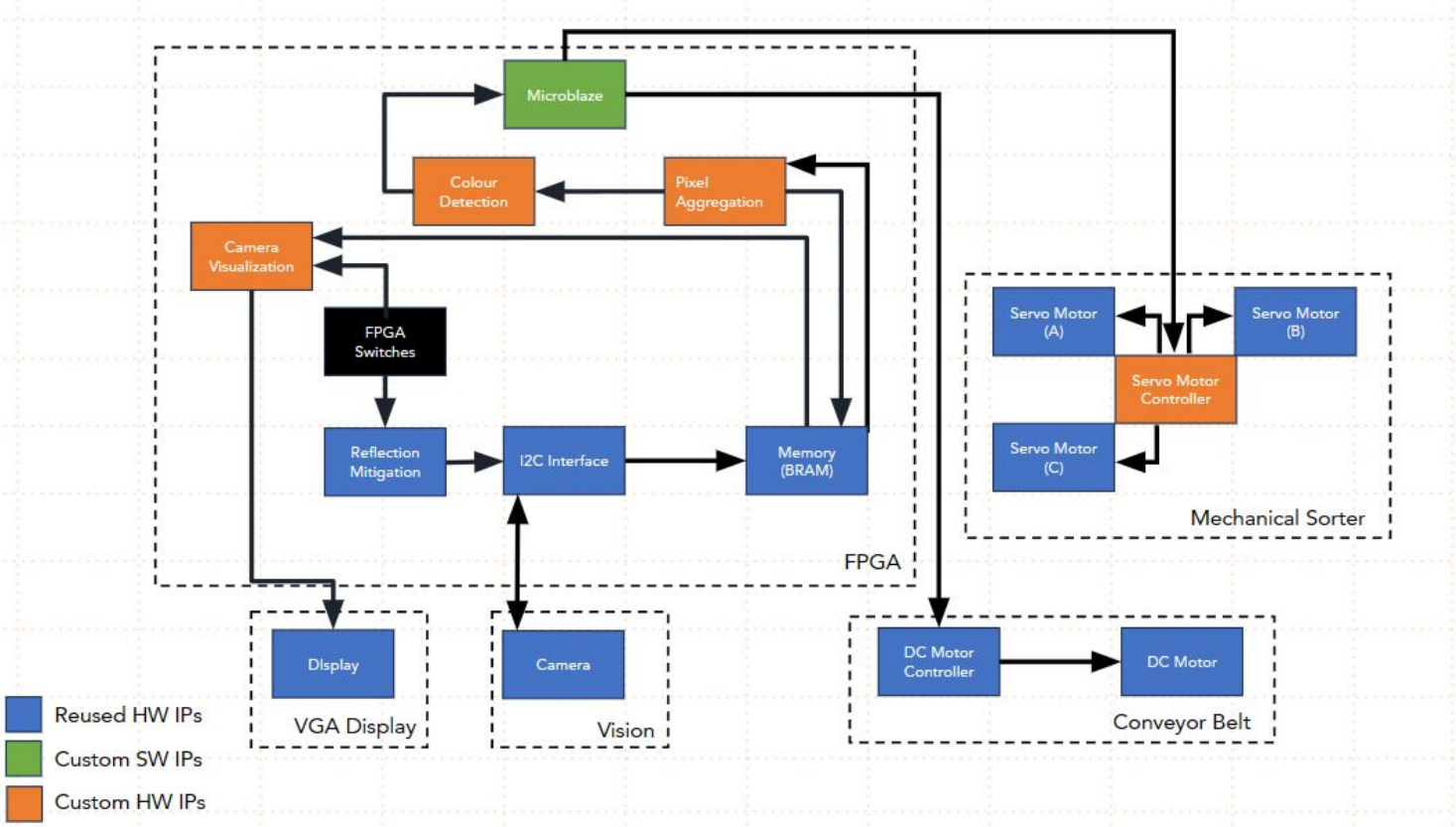
Even if ML is not used:

- FPGAs are power-efficient
- Reprogrammability is useful when iterating on algorithm

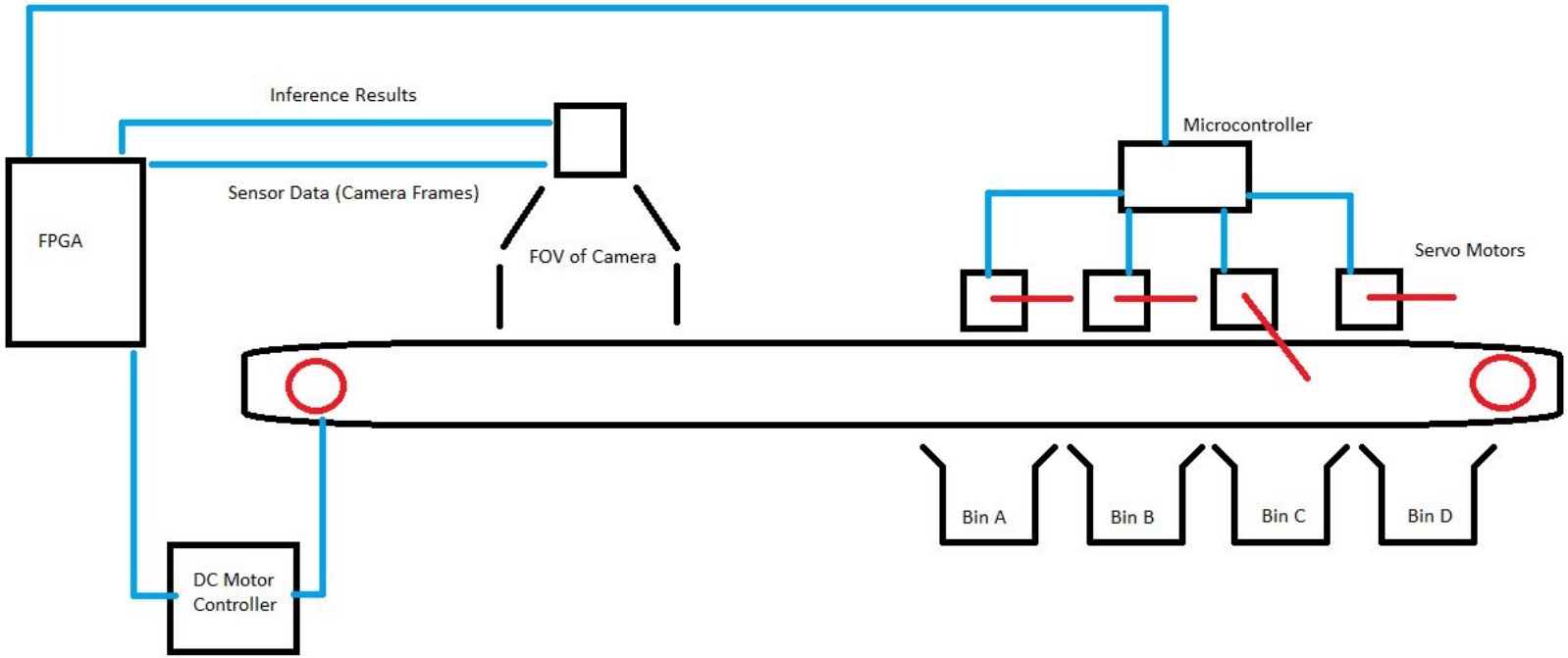
Initial Goal



Final Implementation



Final Implementation - Step by Step



Camera + VGA

Camera communicates using I2C

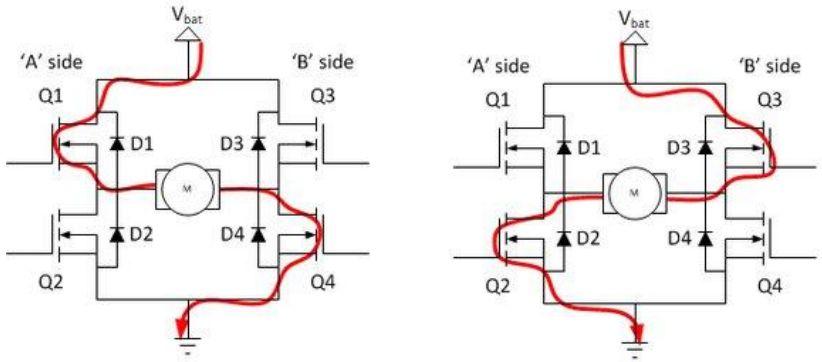
- All communication is done in hardware
- Synchronization is done using SIOC
- SIOD is used for bidirectional data transfer
- Camera is initialized using parameter value configurations stored in registers

VGA output

- Fully handled in hardware
- All pixel indices are looped through per frame
- Pixel value is read from BRAM and written to display individually

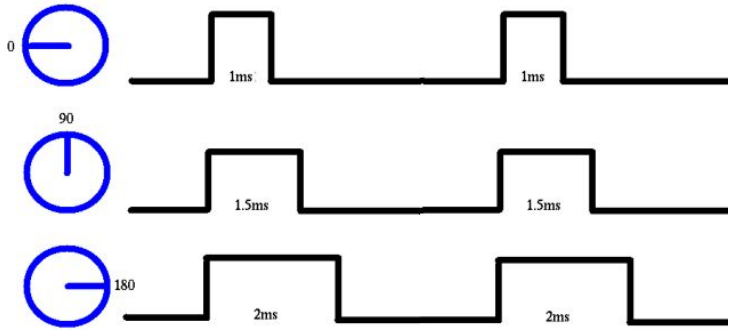
Peripherals - Belt Motor Control

- Controlled using an external H-bridge circuit.
- By sending either a 1 or 0 to the ‘A’ side or ‘B’ side, we can control the motor direction.
- By pulsing the motor enable pin, we can control speed.



Peripherals - Servo Motor Control

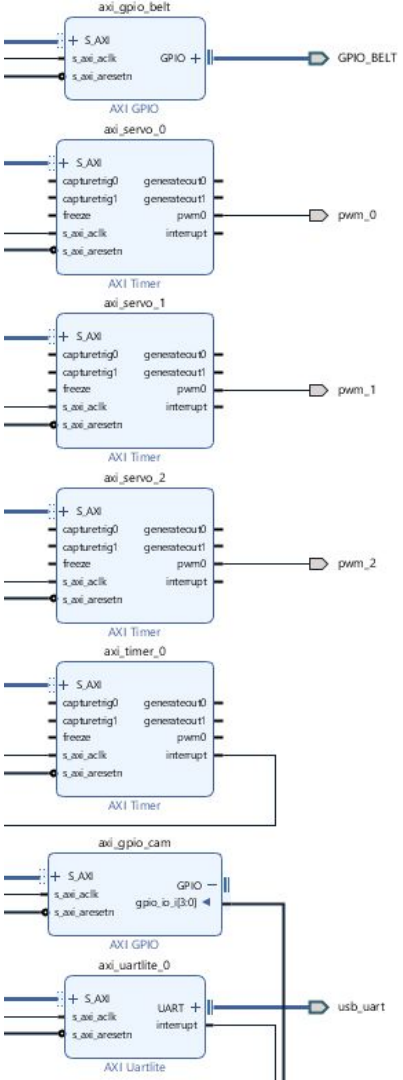
- Controlled using a 50 Hz PWM signal.
- The position of the motor is directly related to the pulse width of the PWM signal.
- An external buck converter circuit is used to step down the main power supply voltage from 24V to 7V to operate the servo motors.



Microblaze

Interface with peripherals through a series of AXI connections

- Belt Control (AXI GPIO)
- Camera Detection (AXI GPIO)
- Servo PWM Signals (AXI Timers)
- Delay Signals (AXI Timers)



Microblaze

Software algorithm to implement sorting sequence:

1. Drive belt forward to move object into frame, then stop
2. Read and average camera detection results
3. Drive and configure servo PWM signals to block & unblock accordingly
4. Drive belt forward to move object into appropriate bin

Custom IP - Decision-making Algorithm

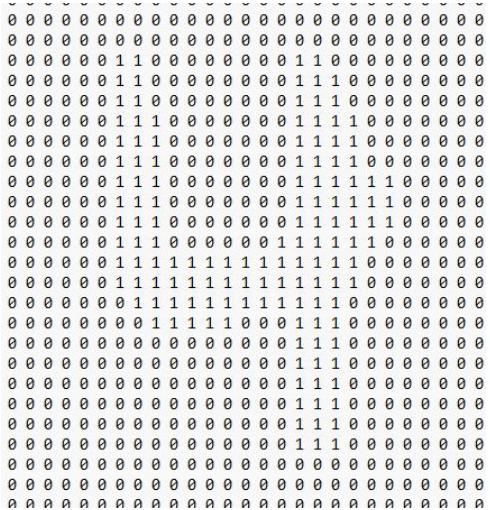
- Reads frames from BRAM frame buffer
- Noise mitigation
  - Aggregates pixel values from BRAM over multiple clock cycles
  - 3 registers used to store intermediate sum values for R, G, B channels
  - Removes LSBs
- Colour Detection
  - Compares relative intensities of colour channels to determine colour combination
  - E.g. Purple is 50% R + 50% B + 0% G

Project Complexity

Component	Complexity Points
VGA Output - No Microblaze	1
OV7670 Camera	1
Software Algorithm: Microblaze + Neural Network Training*	0.6
DC Motor Controller + Servos	0.6
IP core: Decision-making algorithm	0.6
Visualization Tools	0.25
Total	**4.05**

Neural Network - Alternative One

- Dense neural network layers
  - Activation and Softmax functions were implemented using LUTs
  - Training done in Python and weights saved into files
  - Weights loaded onto hardware at reset
- Testing in Python and Vivado
  - Model inputs were 32x32 black and white RAW images
  - Model performed with ~90% accuracy on MNIST in Vivado simulation



Neural Network - Alternative Two

- High-level Synthesis
  - Python library hls4ml
  - Created Tensorflow model and compiled C code for Vivado HLS
  - Result: Custom CNN IP using AXI Stream for inputs and outputs
- Testing in Python and Vivado
  - Model inputs were 120x160 grayscale JPG images
  - Tested in simulation
  - Unable to test on hardware due to time constraints

Descoping from the Neural Network

Difficulties:

- First model failed to run correctly on hardware
  - Errors in how the weights are read
- Second model was too complex (DMA, control signals, etc.)
  - Few more weeks required for integration and testing

Alternative Design:

- Non-ML algorithm for basic image classification
- VGA output was added to compensate for diminished algorithm complexity

Camera Issues

First Camera (OV5642):

+ Lower image noise and higher resolution
+ Optimal for ML tasks
− Lacking documentation and using both SPI and I2C

Alternative Camera (OV7670):

- Switched after Mid-Project Demo
- Quick progress allowed for stretch goal completion (VGA display of camera output)

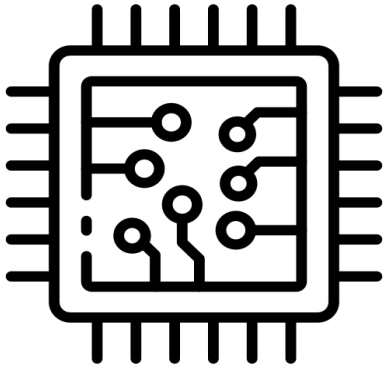
Next steps

NN Alternative One:

- ‘Hard coding’ model weights

NN Alternative Two:

- Test on hardware
- AXI DMA



Questions + Demo

Demo

- Since we have 3 paddles available, we've set the configuration to classify 3 colours:
  - Blue
  - Green
  - White
- Testing performance for highly-reflective objects
- Demonstrating camera sensitivity visualization