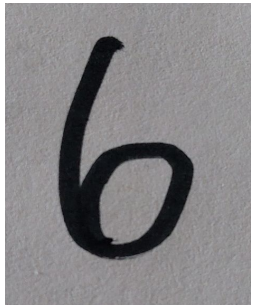
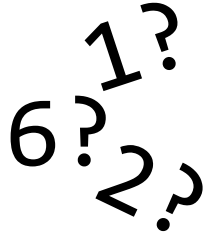
Hardware Image Processing





ECE532 Final Presentation
Group 6

The Team

Frank

Cyrus

Savo

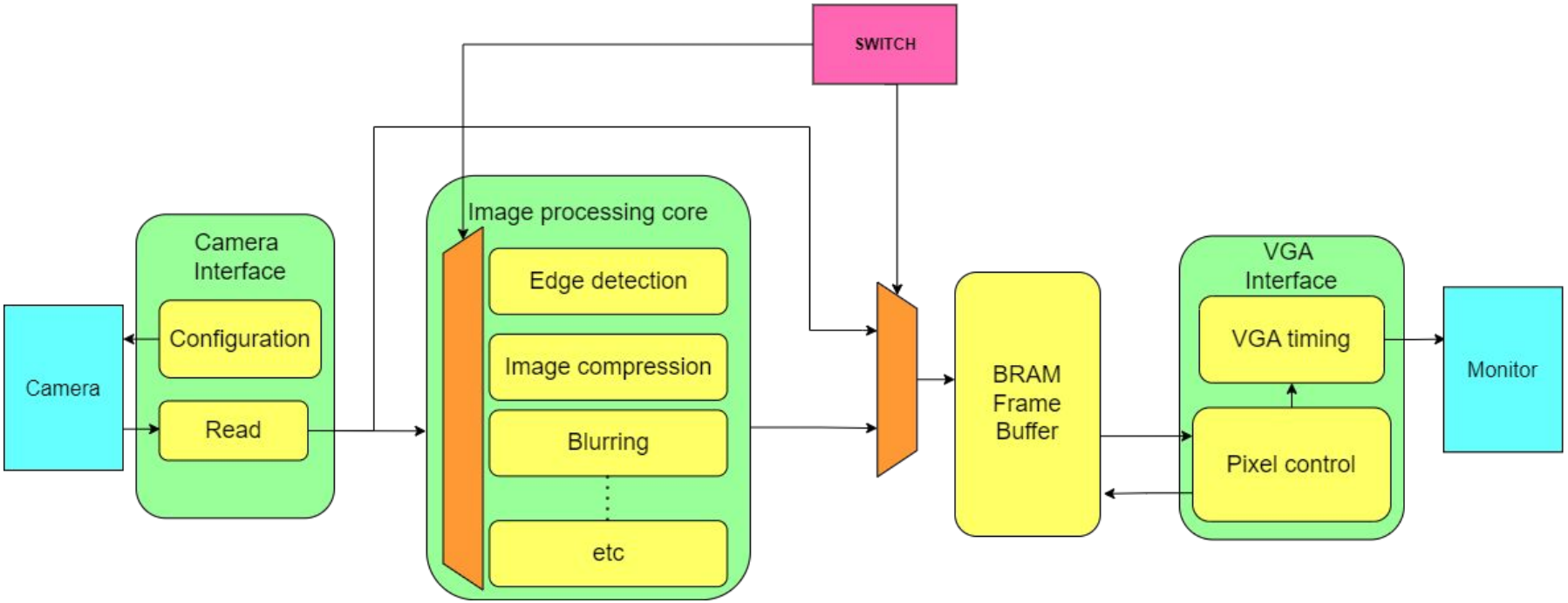
Richard

Initial Goals vs Final Project

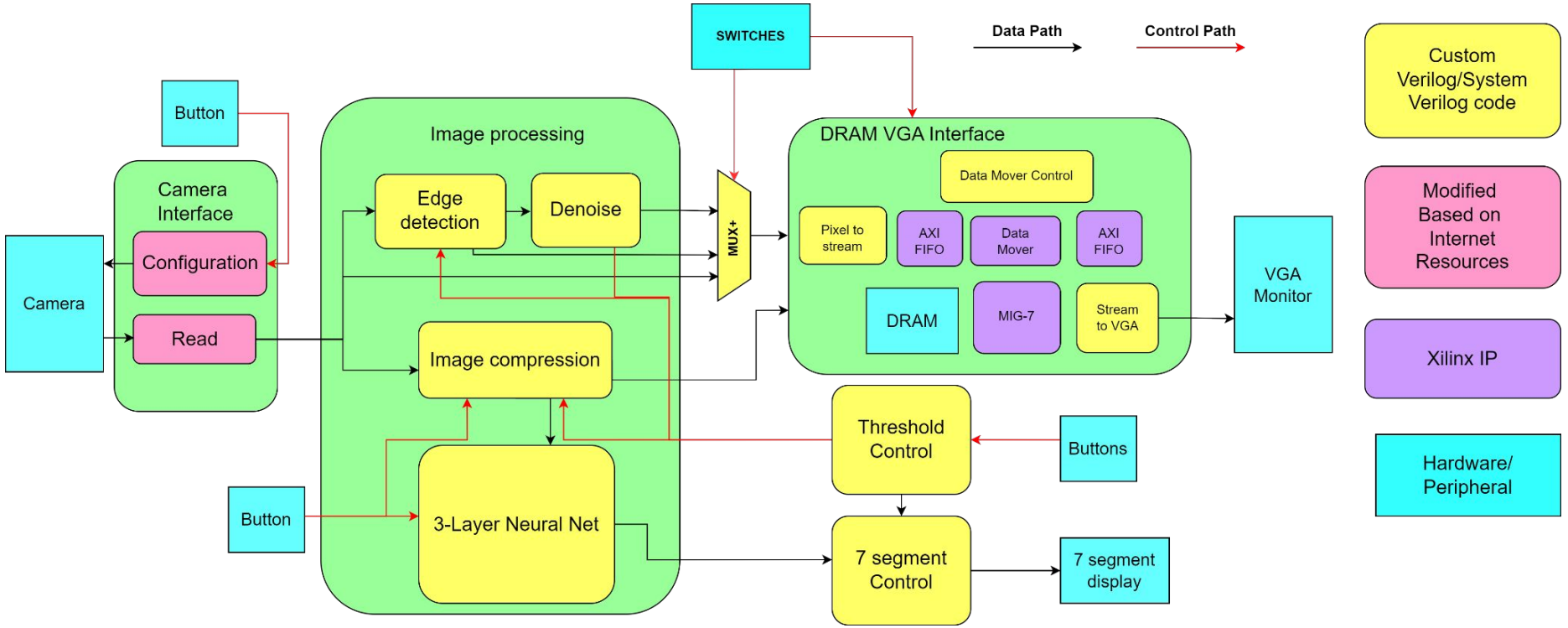
- Reliable image processing pipeline
- Changeable kernels
- Customizable image processing settings

- Reliable image processing pipeline
- Changeable kernels
- Customizable image processing settings
- Digit recognition based on neural network
- Purely hardware !

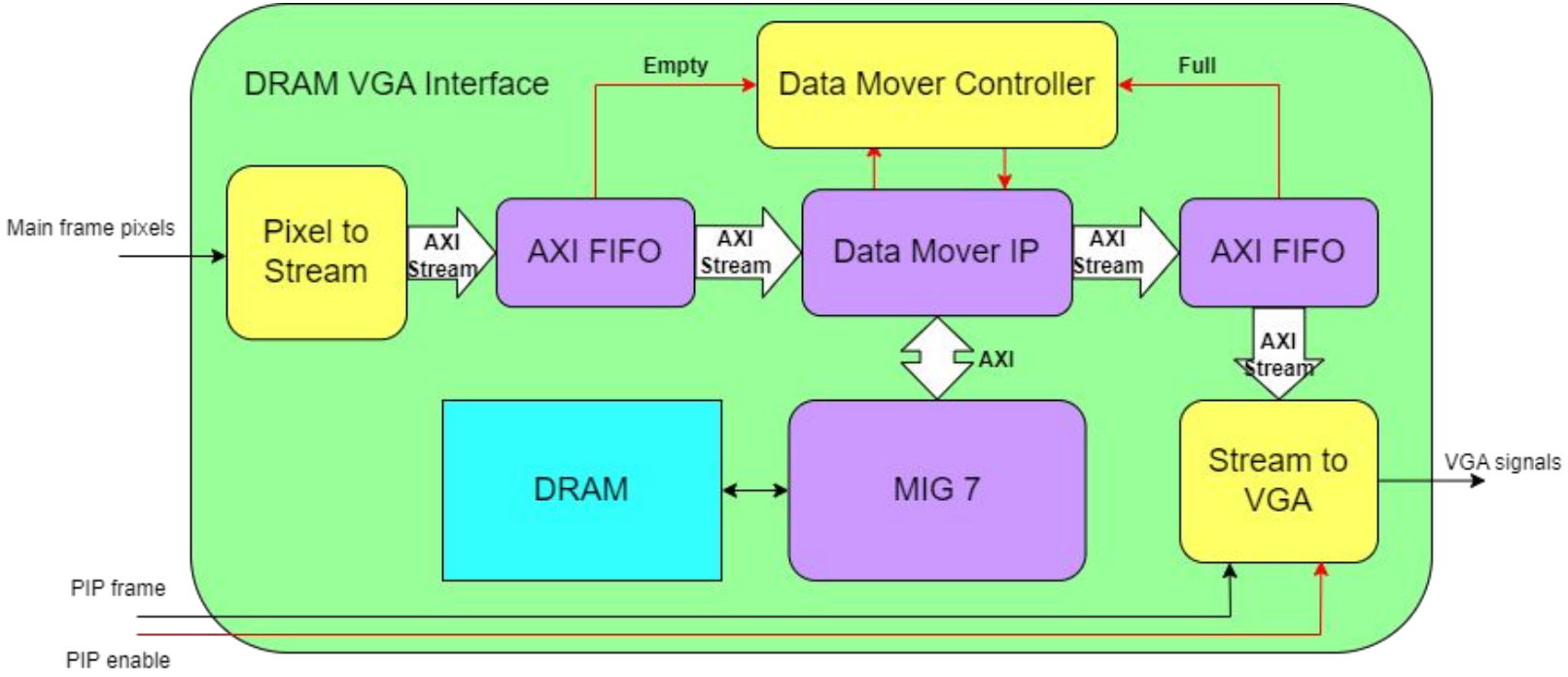
Original Block Diagram (Midproject)



Final Block Diagram



Final Block Diagram –DRAM VGA Interface



Challenges we faced

Data Mover IP Bugs

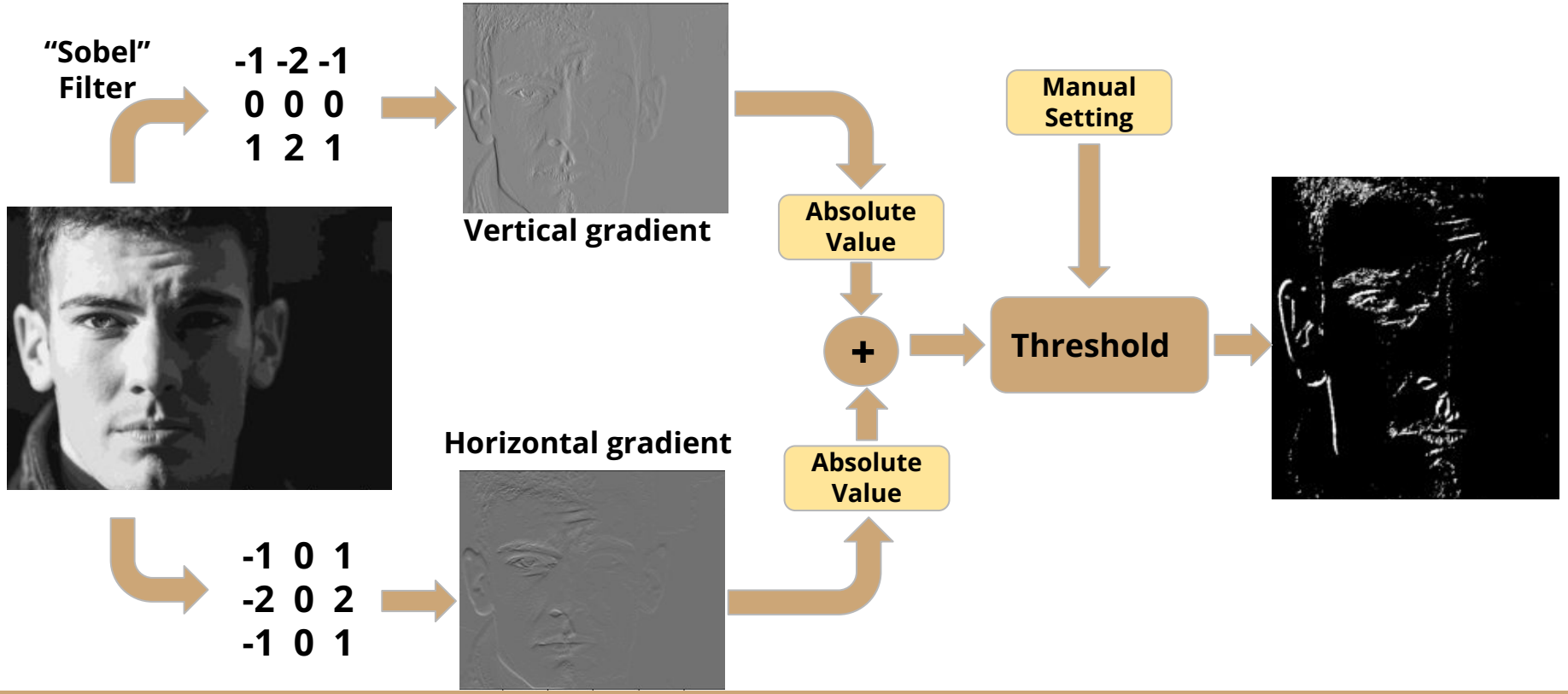
- Not writing data to a continuous region
- Internal buffer on S2MM for two stream packets

0x80000000	0FFF0FFF	0FFF0FFF	AF45C3C3	6697AEE7
0x80000010	0FFF0FFF	0FFF0FFF	65840005	2441AA0C
0X80000020	0FFF0FFF	0FFF0FFF	AD487D2A	080EF016

Accuracy of ML Techniques

- Training Dataset was not taken from our camera + FPGA compression algorithm
- 86% accuracy on test dataset in python does NOT mean 86% accuracy on our system
  - Highly sensitive to position and orientation of number
- More complex Model required to see reasonable accuracy on FPGA

In Depth System Discussion- Edge Detection



In Depth System Discussion- Denoise

The Algorithm:

For each Pixel in the image:

Check 8 neighboring pixels

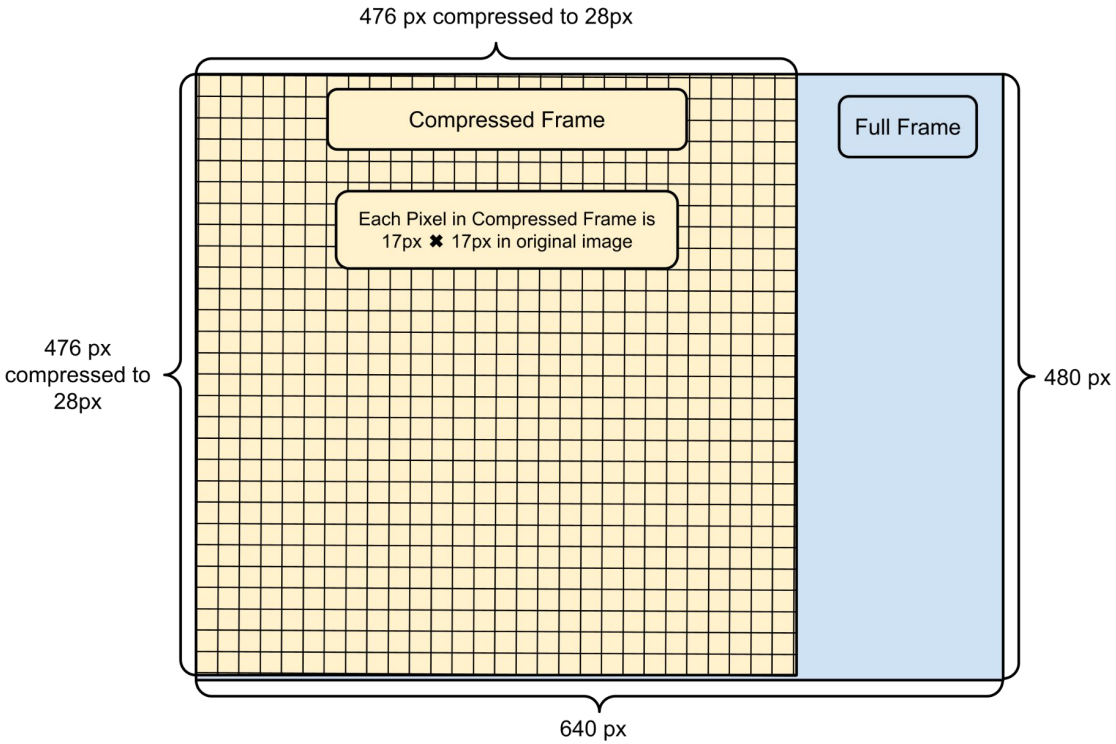
If at least THRESHOLD # neighboring pixels are edges:

Accept this pixel as an edge

Else:

This pixel is “noise”, set to black

In Depth System Discussion- Image Compression



The Algorithm:

For each 17px square in square crop:

> Calculate sum of (15 - pixel intensity) for all pixels in the square

Subtract minimum of sums out of all sums

Scale all sums by a factor so that the maximum is 15

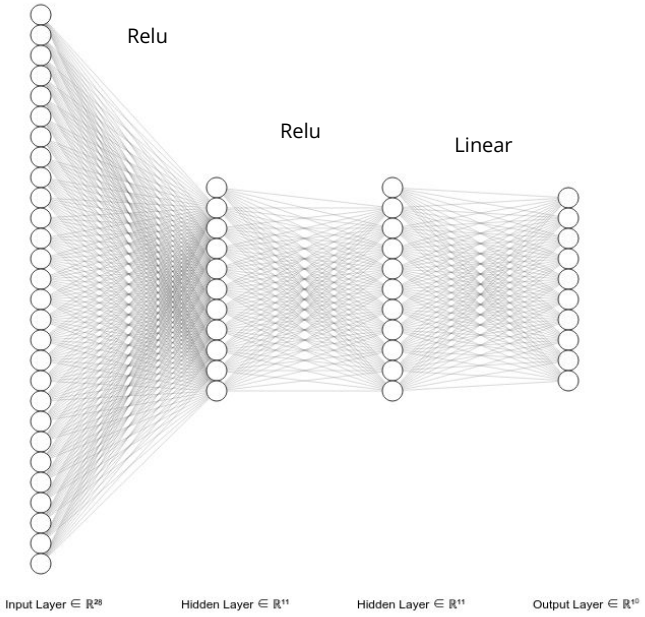
> This is compressed pixel intensity

For all pixel intensities:

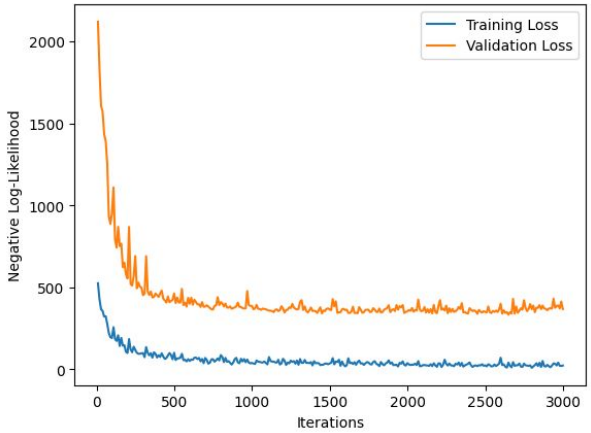
> If pixel compressed pixel intensity < THRESHOLD
>
> Set pixel intensity to 0 (black)

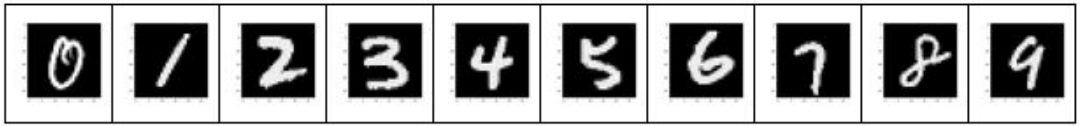
In Depth System Discussion- 3 Layers Neural Net

(1x784) x (784x10) x (10x10) x (10x10)

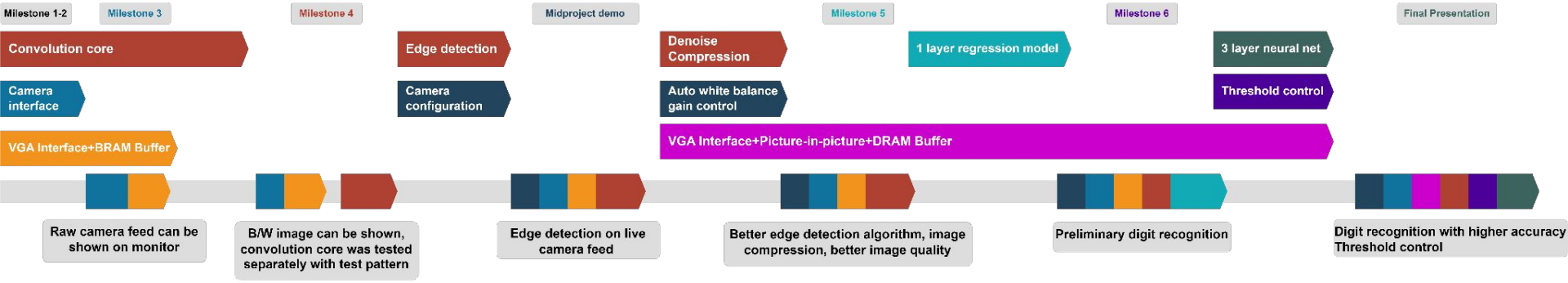


- Linear neural network with biasing
- Trained using gradient descent with backpropagation
- 91.4% accuracy
- 2 hidden layers, 1 output layer





Design Process–A step by step approach



The Final Product

Resource Utilization

- LUTs: 52%
  - Mostly as Logic
  - Also as distributed RAM for pixel buffers
- Flip Flops: 18%
- DSP: 30%
  - Used for Matrix Algebra in Neural Net Classification
- BRAM: 2%
  - Almost No BRAM used; frames stored in DRAM off-chip.

Criterion	Goal
Throughput	10+ FPS ✓ Full 30 FPS
Latency	<200 ms ✓ ~33ms
Reliability	No dropped connections No pixel glitches ✓

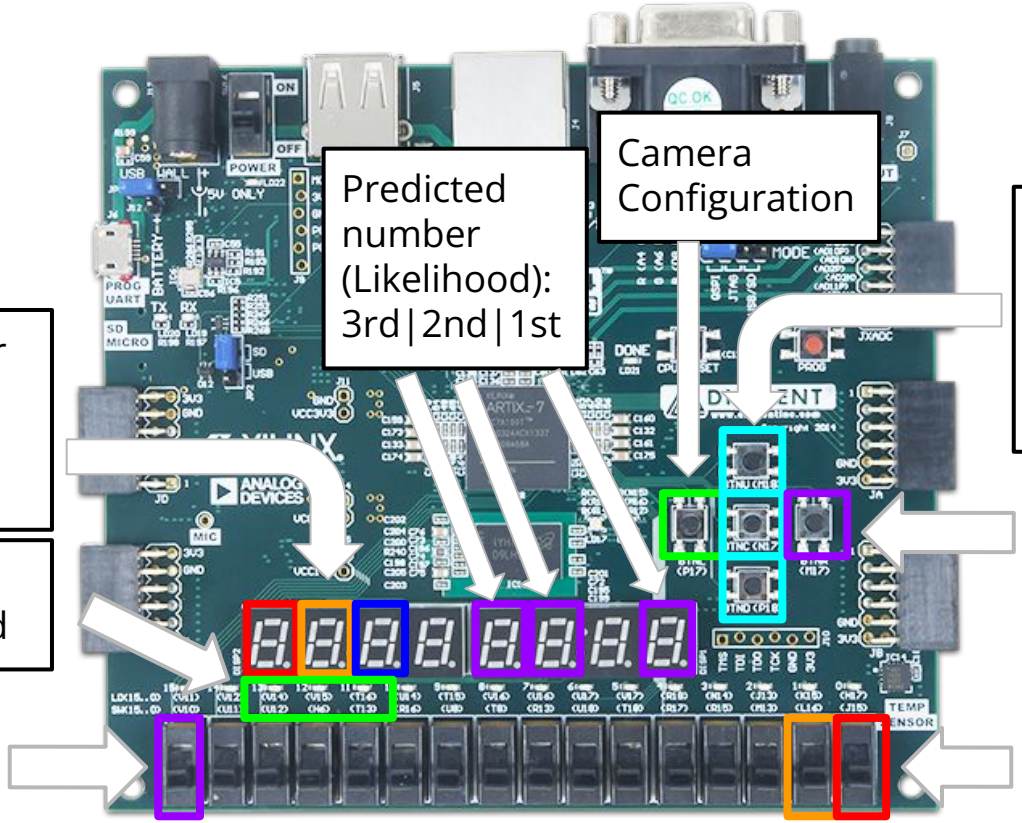
Entirely Hardware, no Microblaze!

Key Takeaways

- (AXI) Streams are powerful
- Properly abiding by interface standards speeds up integration time
- There are many ways to optimize a digital system depending on application
  - DSP vs LUT math
  - Distributed memory, BRAM, DRAM
- Using block diagrams and hierarchies make integration easier
  - Add RTL directly into block diagrams, and wrap systemverilog in verilog code to integrate systemverilog in block diagrams

Demo

Controls



Threshold values for
Edge detection
Denoise
Compression

Leds indicating the value being changed

Toggle switch for picture in picture feature

Predicted number (Likelihood): 3rd|2nd|1st

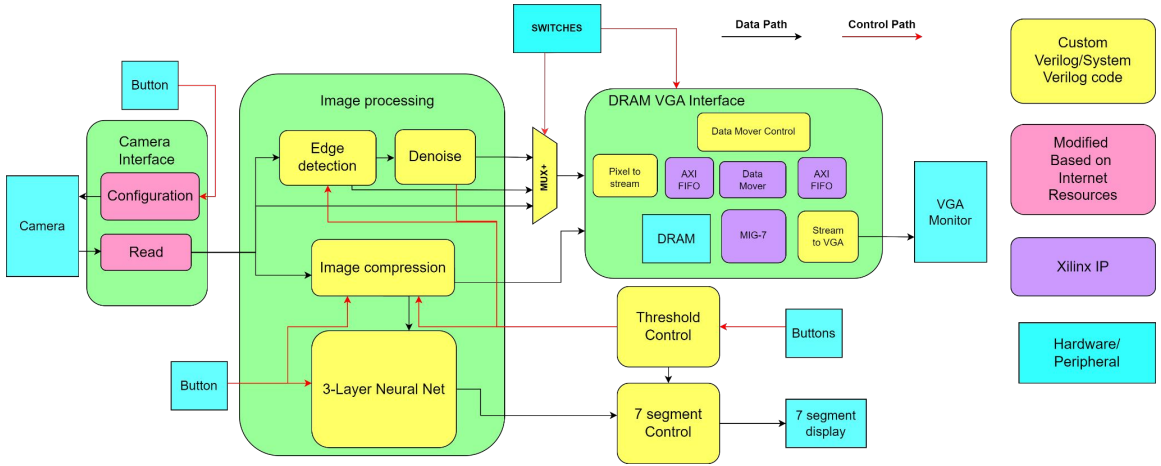
Camera Configuration

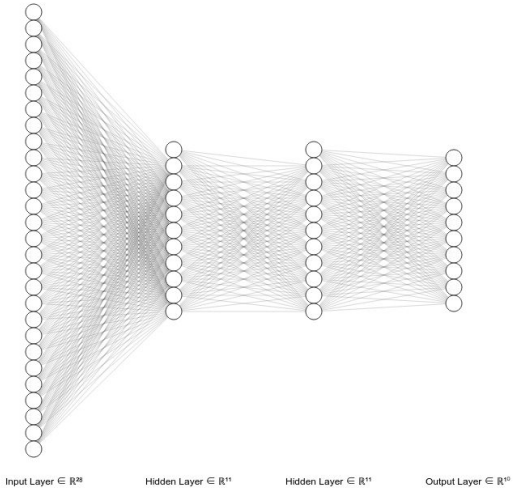
Buttons for changing threshold
Up:Increment by 1
Centre:Change modes
Down:Decrement by 1

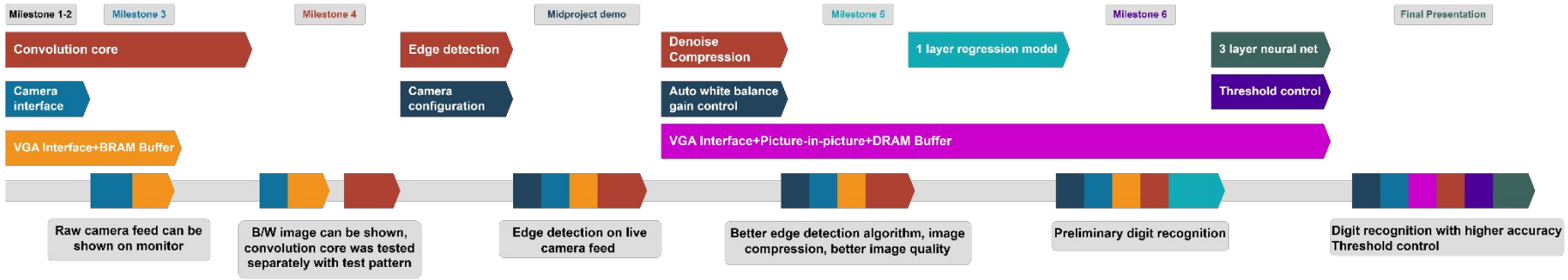
Image Capturing Button

Toggle switches for
Edge detection
Denoise

Q&A


SWITCHES
Data Path
Control Path
Button
Camera Interface
Configuration
Read
Camera
Image processing
Edge detection
Denoise
Image compression
3-Layer Neural Net
Button
MUX+
DRAM VGA Interface
Data Mover Control
Pixel to stream
AXI FIFO
Data Mover
AXI FIFO
DRAM
MIG-7
Stream to VGA
VGA Monitor
Threshold Control
Buttons
7 segment Control
7 segment display
Custom Verilog/System Verilog code
Modified Based on Internet Resources
Xilinx IP
Hardware/ Peripheral


Input Layer ∈ ℝ²⁸
Hidden Layer ∈ ℝ¹¹
Hidden Layer ∈ ℝ¹¹
Output Layer ∈ ℝ¹⁰


Milestone 1-2
Milestone 3
Milestone 4
Midproject demo
Milestone 5
Milestone 6
Final Presentation
Convolution core
Camera interface
VGA Interface+BRAM Buffer
Edge detection
Camera configuration
Denoise Compression
Auto white balance gain control
VGA Interface+Picture-in-picture+DRAM Buffer
1 layer regression model
3 layer neural net
Threshold control
Raw camera feed can be shown on monitor
B/W image can be shown, convolution core was tested separately with test pattern
Edge detection on live camera feed
Better edge detection algorithm, image compression, better image quality
Preliminary digit recognition
Digit recognition with higher accuracy Threshold control