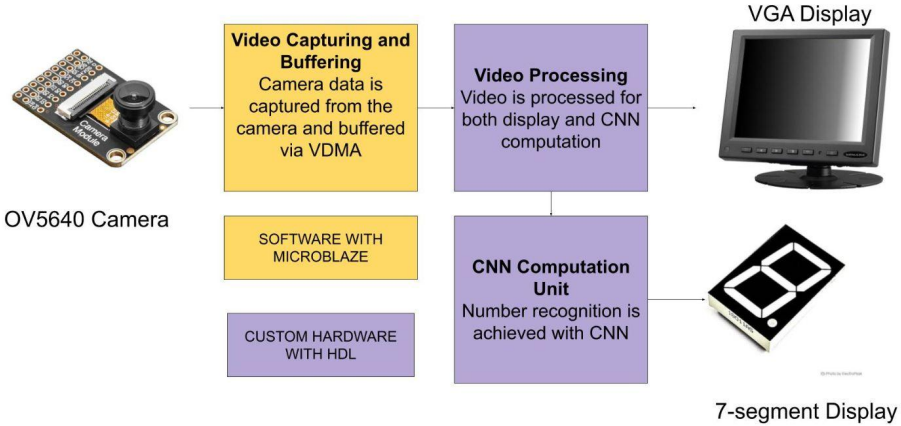
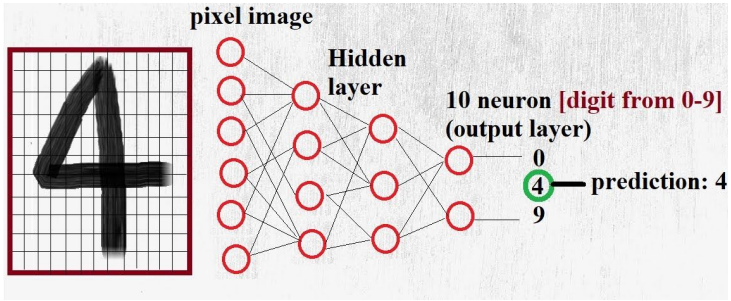
CNN-based Number Recognition Using FPGA Acceleration

ECE 532 - Group 6

Chen Chen
Lily Li
Bowen Liu
Ruoyi Xie

High-Level Project Description

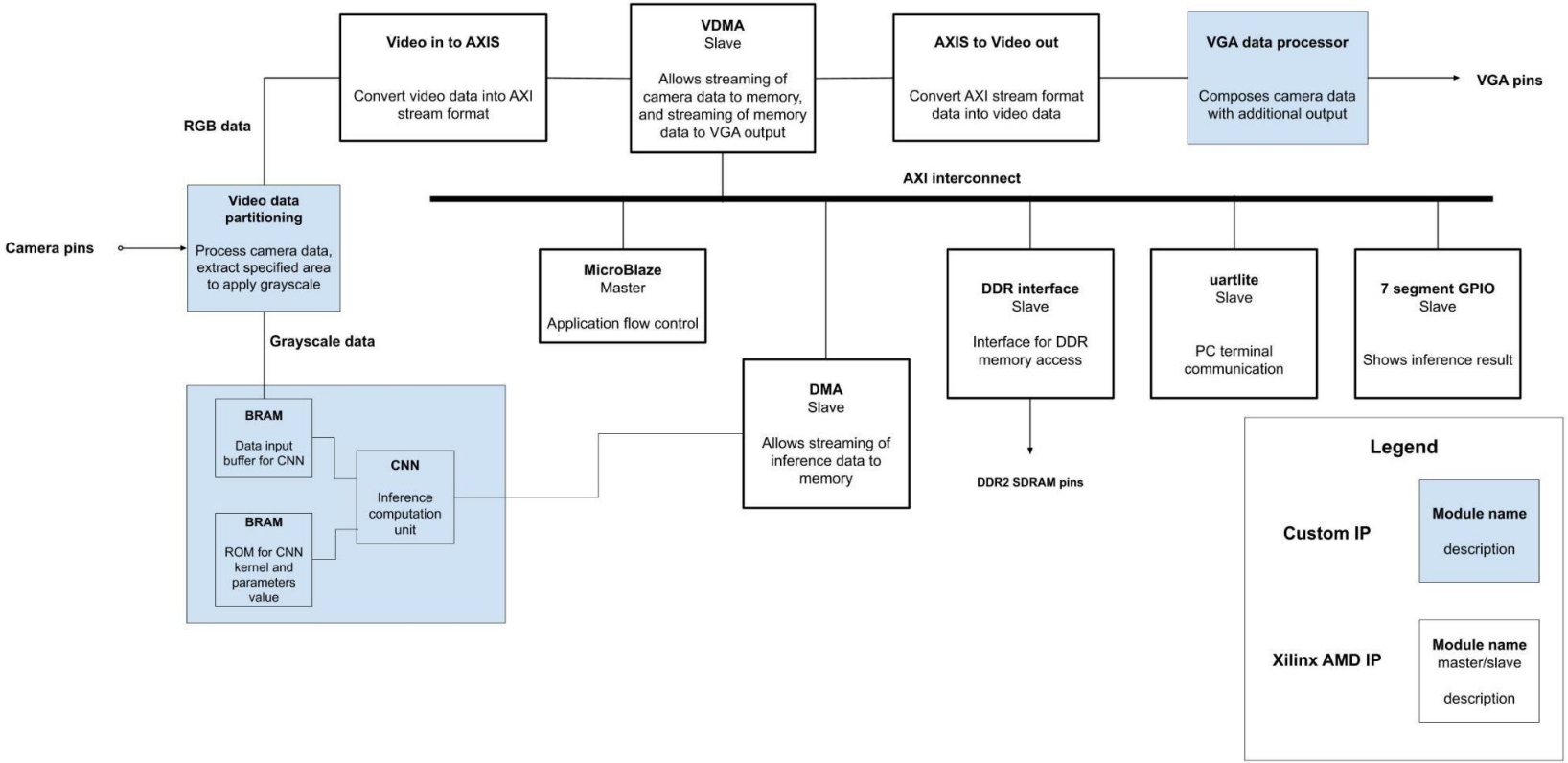




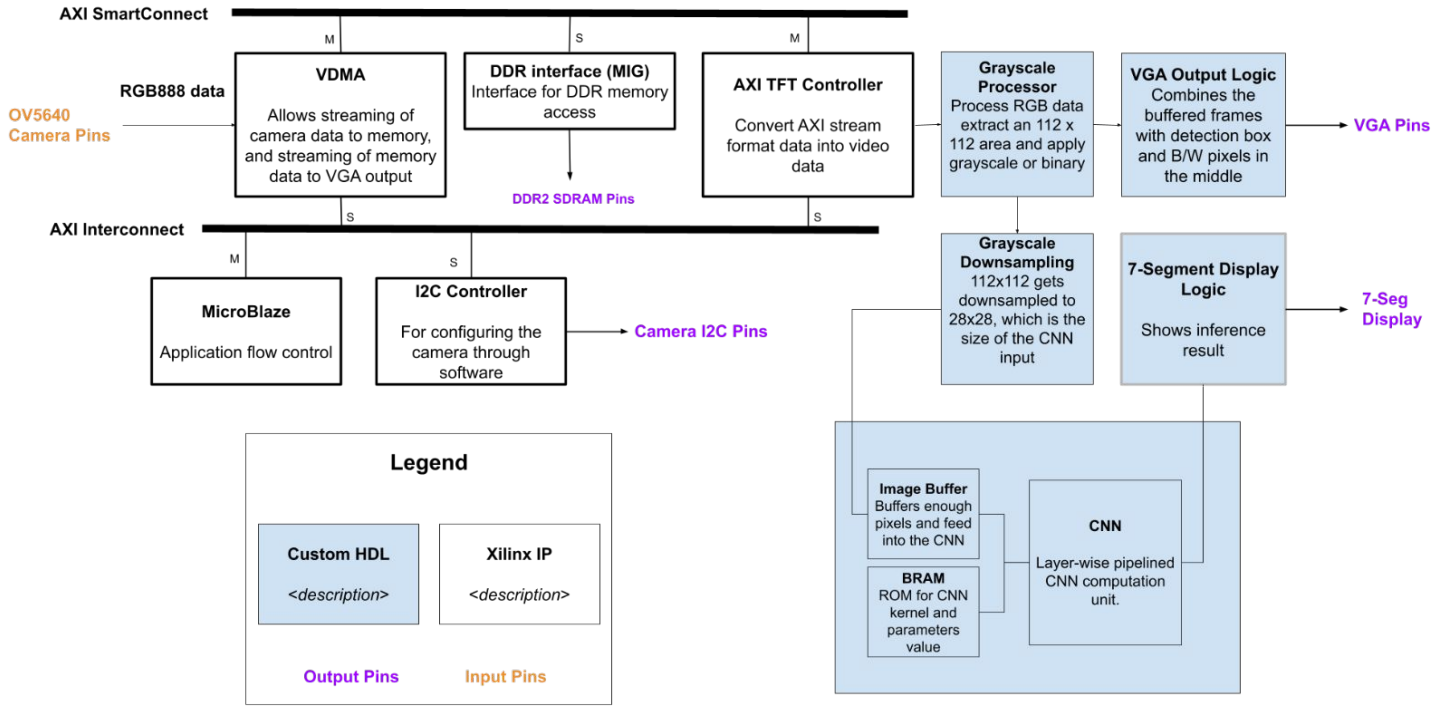
Goal: Deploy a CNN on a Nexys 4 DDR FPGA for efficient handwritten digit recognition.

- Camera captures a selected area covering the handwritten digit.
- The MicroBlaze subsystem manages data transfer.
- Custom HDL codes are written to target signal processing and acceleration.
- Captured images will be displayed via VGA
- Results will be displayed on 7-segment display

Proposed System Design——Block Diagram



Final System Design — Block Diagram



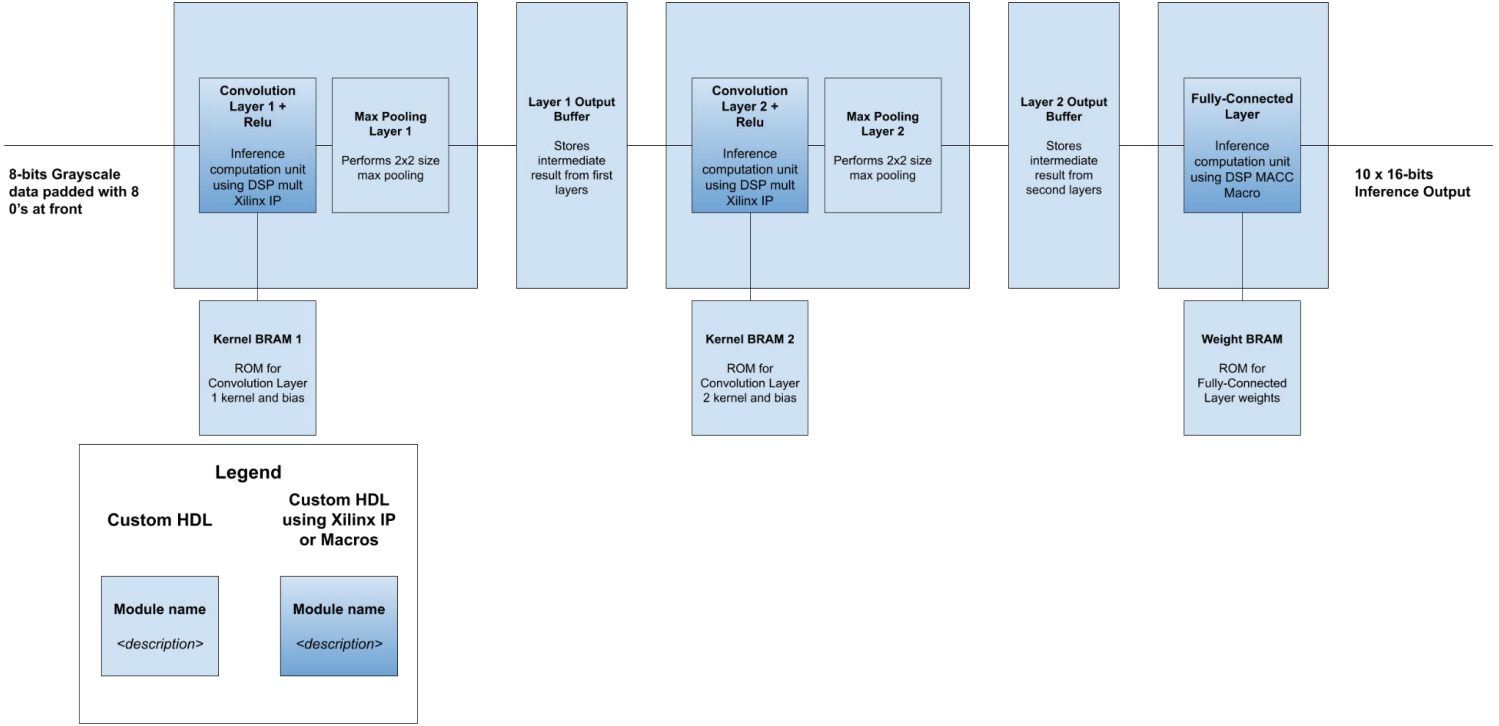
Implementation of the Video Capturing and Output Unit

- I2C settings are set using microblaze
- RGB888 camera data is combined to AXI Stream, and VDMA is used to buffer in DDR for smooth transfer
- AXI Stream data is converted to video data through TFT Controller
- Video data within the detection box is converted to grayscale to be displayed on screen
- A switch is used to change between grayscale and black and white display
- 7-Segment display is used to display the CNN output. It displays once the CNN computation completes and the maximum possibility is positive.

Implementation of the Video Processing Unit

- Two counters are implemented to keep track of the pixel location.
- The pixel is captured in the 112x112 detection box. Each new frame is indicated by a pulse.
- Data is turned into 8-bit grayscale or binary.
- The 112x112 data is downsampled to 28x28 (choose one pixel out of 4x4 pixels).
- The 28x28 data is stored in a buffer and a valid signal will be asserted to the CNN when it is full.
- Data are fed into the CNN 5 pixels at a time when addresses are received.

Implementation of the CNN



Implementation of the CNN

- CNN architecture:
  - Conv1: 28x28 inputs, 5x5 kernel, 1 input channel & 4 output channels
  - Conv2: 12x12 inputs, 5x5 kernel, 4 input channels & 4 output channels
  - Fully-Connected: 64 inputs, 10 outputs
  - total 1158 parameters
- Pytorch floating-point 32 bits model converted to 16 bits fixed point, using quantization specific for each layer
- ~5% mean percentage error
- Inference time: 150 us @ 25MHz

Implementation of the CNN

- handshaking between computation layers and buffers, signaling start and end of operations
- Bram buffers used in between each layer, allow pipelined computation for each layer to support (potentially) higher throughput and frame rate

Difficulty Score

	Complexity
VGA output using MicroBlaze	0.75
Use of 7-seg Displays	0.20
OV5640 Camera Integration	1.00
Video Processing + Buffering between Video and CNN	1.00
Entire CNN Module	2.00
Total	**4.95**

Potential Improvements

- **CNN model**
  - CNN model can be retrained with augmented images that could have various size scaling, position shifting and background noises
  - CNN model can potentially be reduced in size in terms of bitwidth and parameters
  - CNN hardware implementation can be optimized to reduce computation time
  - More powerful CNN models can be deployed for more challenging tasks: letters and numbers, etc.
- **Video quality**
  - Contrast and exposure
  - Latency and refresh rate

Thank you! Questions?