ECE 385 Final Project Whiteboard

Components: [Platform Designer]

- ☐ SPI Controller (Mouse)
- ☐ VGA Graphics Controller
- ☐ NiosII/f
- ☐ Neural Network IP
- ☐ On-Chip RAM/ROM
- ☐ SDRAM
- ☐ PLL
- ☐ SysID/UART

System-Level Organization:

- NiosII/f handles mouse movement.
- Grab character images from screen and send to CNN.
- Display CNN output to screen.

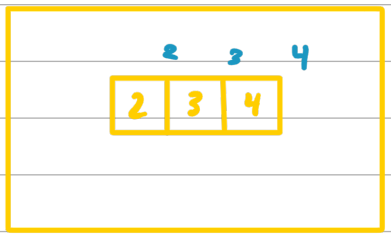
Validation:

- Python/MATLAB to validate image algo and CNN.
- SV Simulations to validate hardware

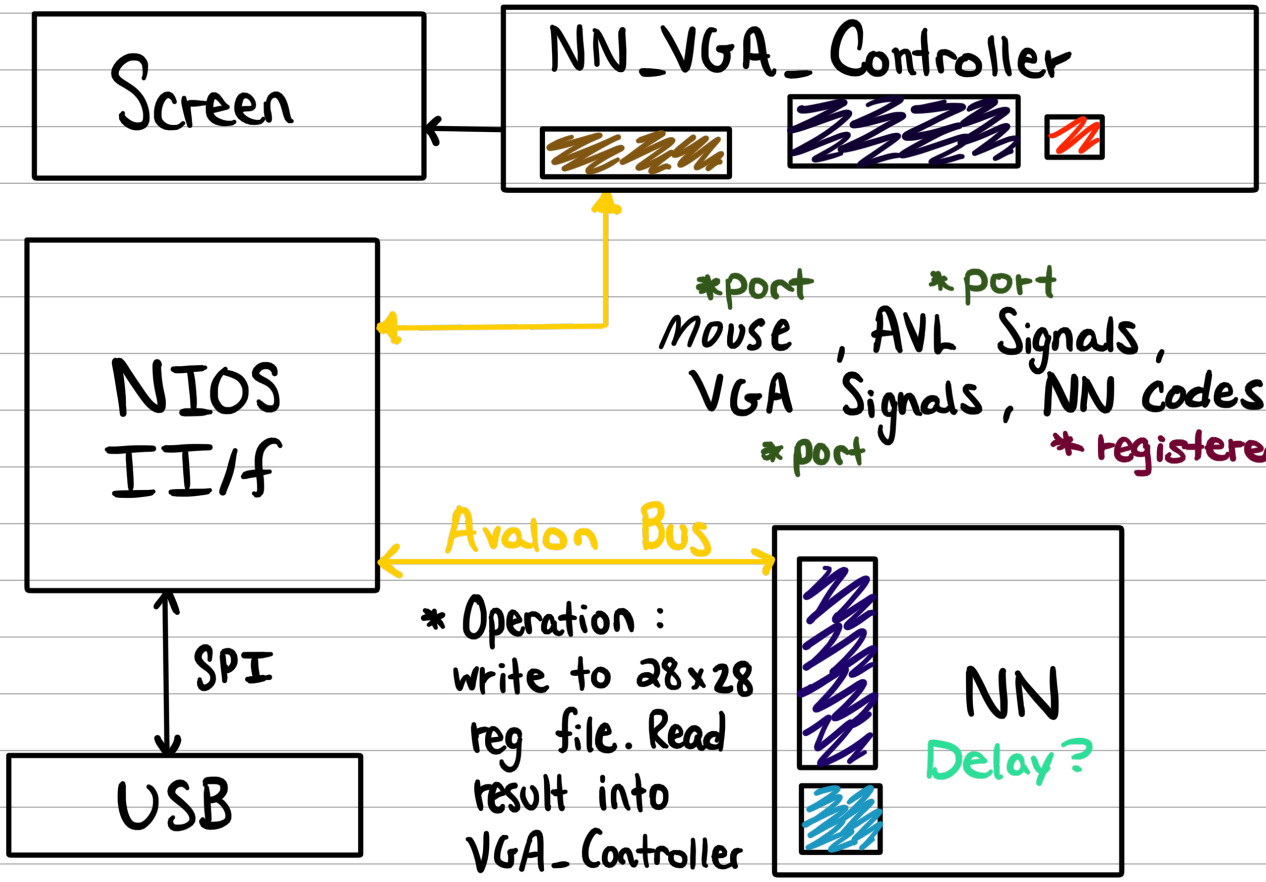
Checkpoints:

- Mouse can draw.
- Can place bounding box on a single character.
- Recognize 1 character @ a time.
- Recognize String of characters
- Compute simple math (2+2)
- Complex expressions (2+2)x2+2
- Add nicer ui ++

Characeters in only a certain region are recognized



System Level Block Diagram:

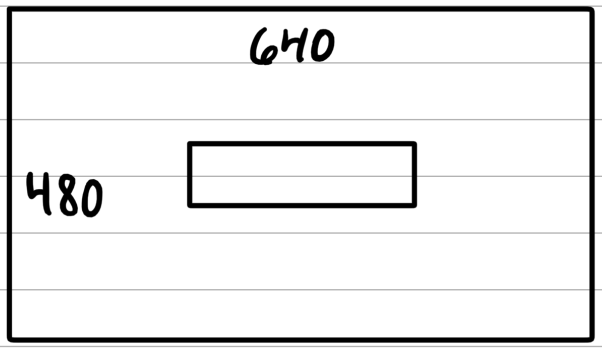


- NN Codes Register File
  x5 16-bit Registers
- 28x28 Register File
  x 28 32-bit Registers
- VRAM
  * Check bit mapping
- 16-bit Output Register
- Blank Signal

Operation:

- Read into NN only allowed during blank.
- Write to screen any time.
- Mouse rendering updates happen locally in VGA Controller.

VRAM Bit Mapping: 32-bit data width



* B & W (1 bit per px): 640 x 480 = 307,200 bits
20 x 480
9600

Grayscale (2 bit per px): 640 x 480 x 2 = 614,400 bits
40 x 480
19200

Grayscale (4 bit per px): 640 x 480 x 4 = 1,228,800 bits
80 x 480
38400

Total bit memory:

1,677,312 bits

Memory Map (Important Ones):

SDRAM:	0x04000000 - 0x07FFFFFF
VGA_interface:	0x08000000 - 0x0800FFFF
NN_interface:	0x08011000 - 0x080110FF
SPI:	0x080111A0 - 0x080111BF

Mouse Information

I/O From NIOS:

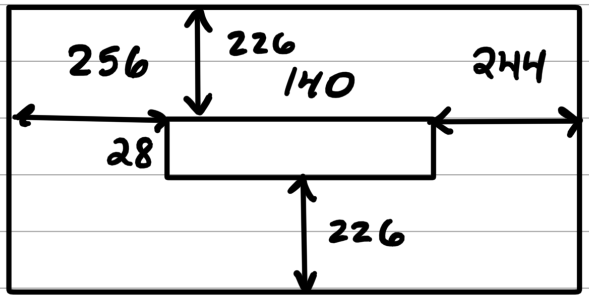
- Xdisplacement PIO
- Ydisplacement PIO

- Small Movement Tolerances:

If (MouseX or mouseY < tolerance)
    don't apply sensitivity
else
    apply sensitivity

Neural Net Pixel Mapping:

28x28 boxes, 5 boxes : 140 x 28 region

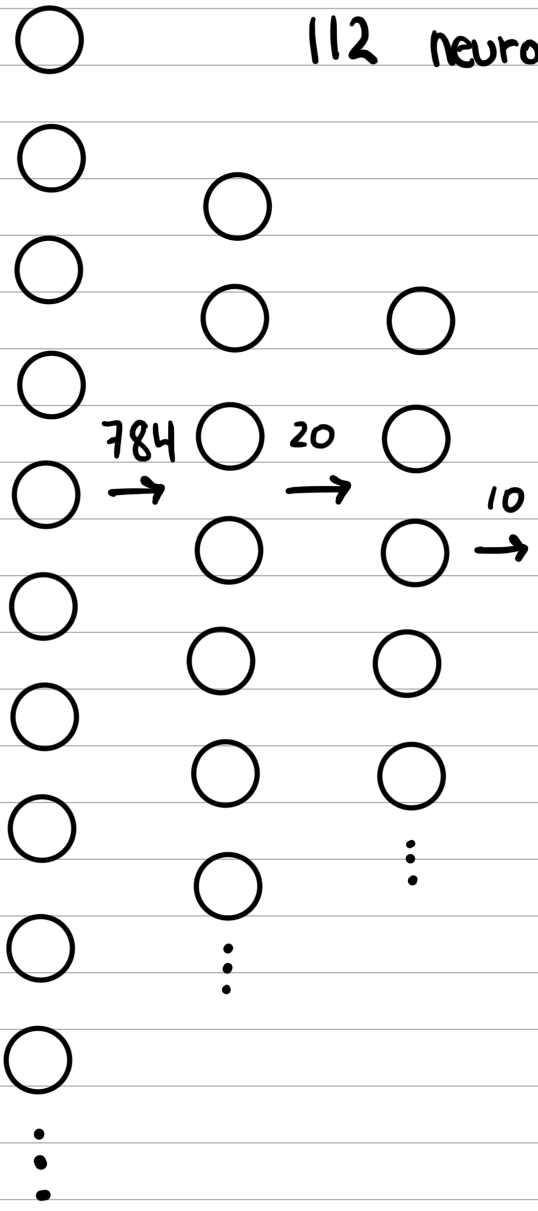


0-639

Start @ Reg 9 in row

Img 1: Idx: [256, 283]
Img 2: Idx: [284, 311]
Img 3: Idx: [312, 339]
Img 4: Idx: [340, 367]
Img 5: Idx: [368, 395]

Neural Network Design Plan:



112 neurons

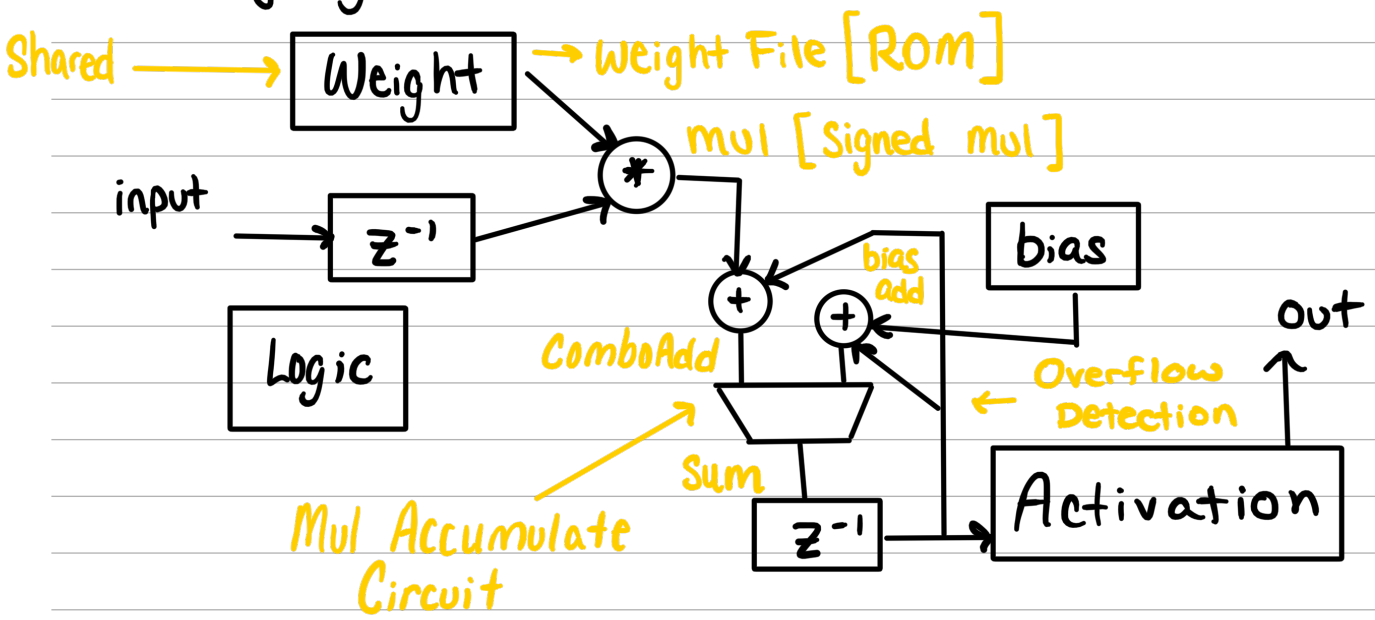
Estimated Resources:

75 LUT
56 FF
2 DSP
1 BRAM

Total Resources:
8,400 LUT
6,272 FF
224 DSP
112 BRAM

Neural Network on FPGA: Sequential

→ Designing a neuron : $\sigma(z) = \frac{1}{1+e^{-z}}$



Mul: [2*datawidth -1 : 0]

- $n_1 * n_2 = m+n+k+j$ bits

- 2 bit sign
- (m-1)(k-1) → integer part
- n+j → fractional part

Activation Function: 16 bit output : ROM

SigmoidSize : memory depth, upper bits of sum

Use python program to generate.

Layer Design:

- Use control logic and shift register to sequentially input data
  → Layer size decreasing

- Input to neural net using image buffer

Output Detection:

- Softmax Function : Software Implementation
- Hardmax Function : Hardware Implementation
  - Remove Processor Interface

MIF:

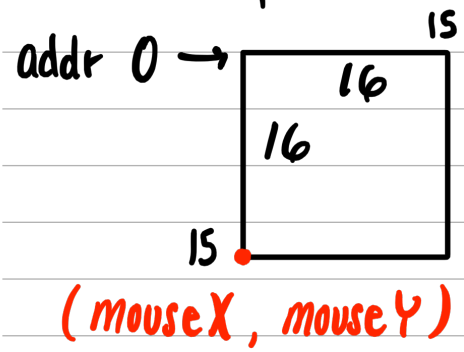
w_ layer#_ neuron#. mif
b_ layer#_ neuron#. mif

include.v → Config file

Generating Weights & biases:

→ Use python Script file

Pencil Sprite: 16 x 16 Sprite



Condition:

Draw X ≥ Mouse X
&& Draw X < MouseX + 16

Draw Y ≤ Mouse Y
&& Draw Y > MouseX - 16

ROM Mapping:

ROM_Addr = (Draw X - mouse X) +

16 * (Draw Y - (mouse Y - 15);

Color Index: 8 colors → 3 bits

Neon Green	x0	R = C	G = F	B = 0
Black	x1	R = 0	G = 0	B = 0
Orange	x2	R = F	G = D	B = 0
Gray	x3	R = 6	G = 6	B = 6
Silver	x4	R = B	G = B	B = B
Tan	x5	R = F	G = E	B = C
Pink	x6	R = F	G = A	B = C
Brown	x7	R = 9	G = 7	B = 4

NN Simulation Test:

#0	E = 7	A = 5	#5	E = 1	A = 1
#1	E = 2	A = 1	#6	E = 4	A = 4
#2	E = 1	A = 1	#7	E = 9	A = 1
#3	E = 0	A = 0	#8	E = 5	A = 1
#4	E = 4	A = 4	#9	E = 9	A = 5

NN ModelSim Validation:

100 Tests 500 Tests 10000 Tests : 1/3 sec

Using Integer width : 7 => 54%

Using Integer width : 5 => 92%, 88.4%

Using Integer width : 4, 3 => 94%, 92%, 91.8%

Using Integer width : 2 => 89%

Must use more Sig memory
=> 10 bits.

~ 33 µs compute time : 33.222856 µs

Python : 0.2875 sec @ Fmax (75 MHz)
0.2215 sec,

For Integration in SoC
after 500 test Cases => 88.4%

System Integration Tasks:

1.) NN interrupt and start ✓

2.) NN hardware validation : Image in SDRAM ✓

3.) Display character Register ✓

4.) Setting Characters from VRAM in SDRAM ✓

Move Box Border + 1

5 6 65 18

01010110 00011000

11110000

01101111 11110101 4

00000110 ce 00110111

1000 1101

8 d 7 3

000[illegible]0[illegible] 0[illegible]00[illegible]

1 B e C

0001 1011 11101100

3 6

00[illegible]0[illegible]0

12 1100 c e 1110 14

00[illegible]0[illegible]

3 7

0011 0111

0000 = 0	→ 0		1000 = 1	→ 8	
0001 = 8	→ 1		1001 = 9	→ 9	
0010 = 4	→ 2		1010 = 5	→ A	
0011 = C	→ 3		1011 = D	→ B	
0100 = 2	→ 4		1100 = 3	→ C	
0101 = A	→ 5		1101 = B	→ D	
0110 = 6	→ 6		1110 = 7	→ E	
0111 = e	→ 7		1111 = F	→ F	