FPGAScope

ECE241 Final Project

Rahil Harit-Singh and Ayan Ali

FPGAScope

ECE241 Final Project

Project Description

Rahil Harit-Singh and Ayan Ali

Our project is a 4-channel oscilloscope.

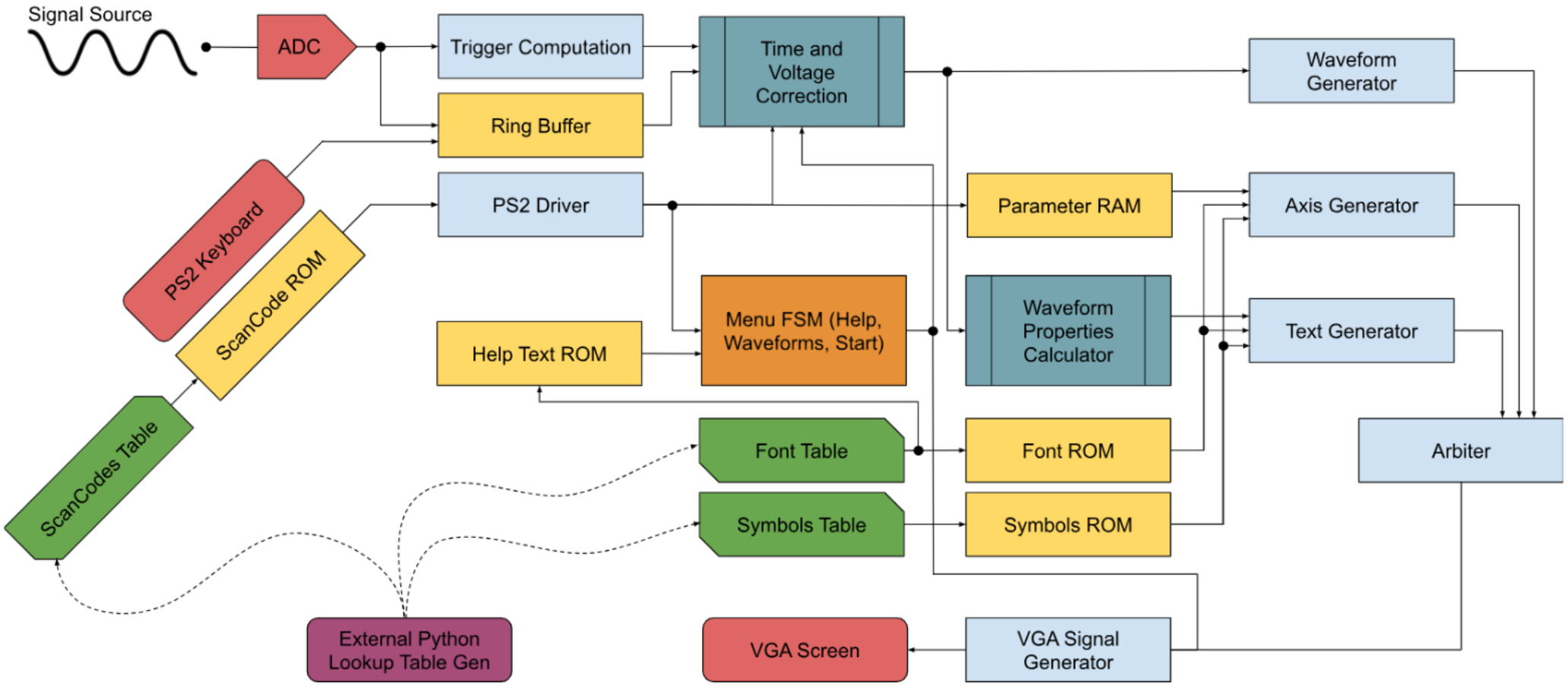
Technical Specifications

- Up to 25kHz wave as the current sample rate of the ADC is 50KSPS/channel.
- Up to 2.5V maximum amplitude with an offset of 1.25V required for every wave, given that the ADC is placed in unipolar.
- Timebase range goes from 5s down to 50µs, voltage scaling available from 50mv to 2V.

Features

- The scope contains a start-menu, a waveform-screen, and a help-screen with the project's keybinds.
- The scope contains live-readouts of Vmax, Vmin, Vp2p, Period, and Frequency.
- The scope contains cursors that are adjustable to gauge voltage and time differentials.
- The scope has an adjustable time-base and voltage scaling alongside independent offsets in both the time and voltage axis for each channel.
- Contains Single, Normal, and Auto Trigger types with both falling and rising variants.

General Block Diagram

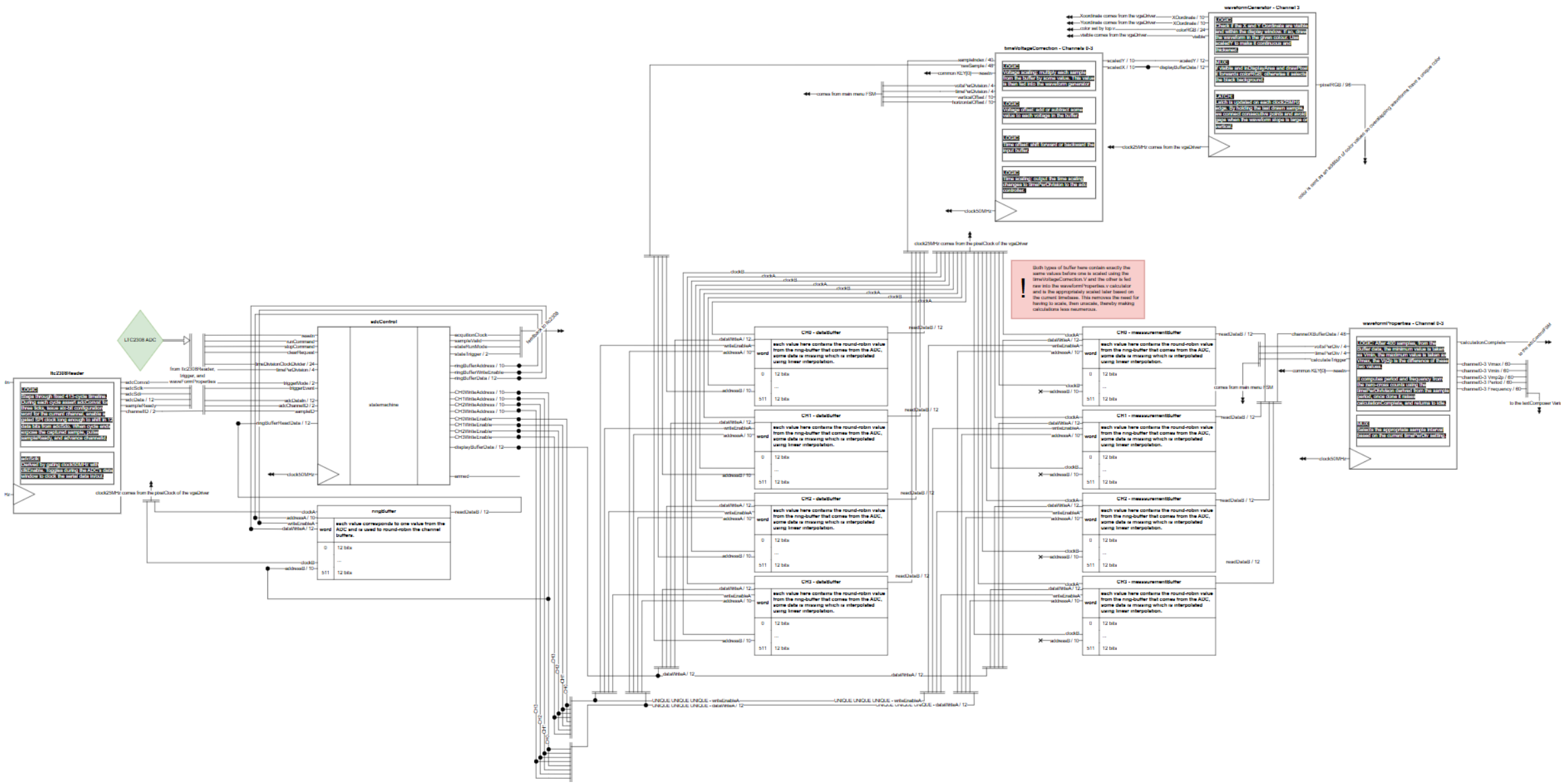


FPGAScope

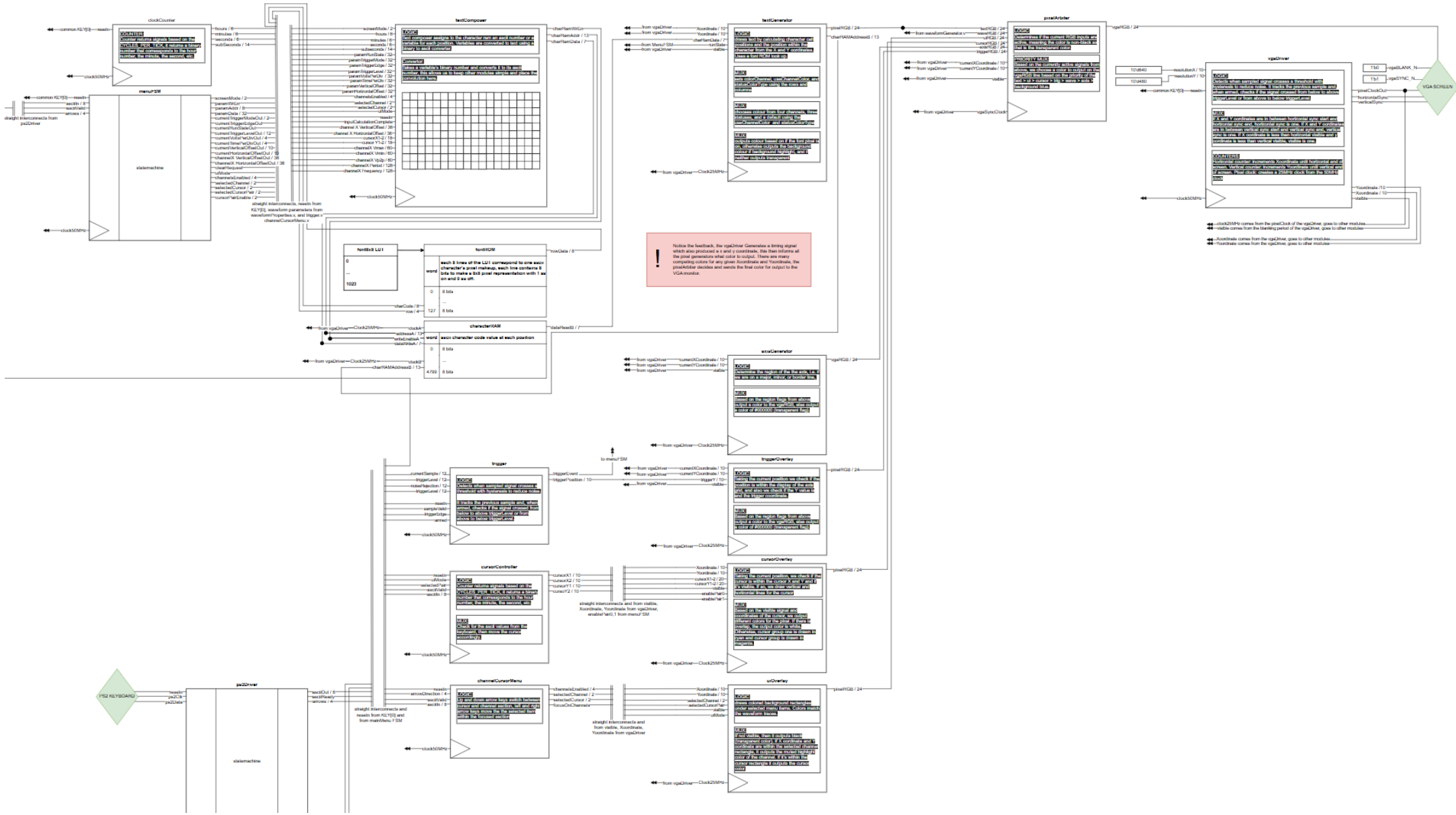
ECE241 Final Project

Project Specifics

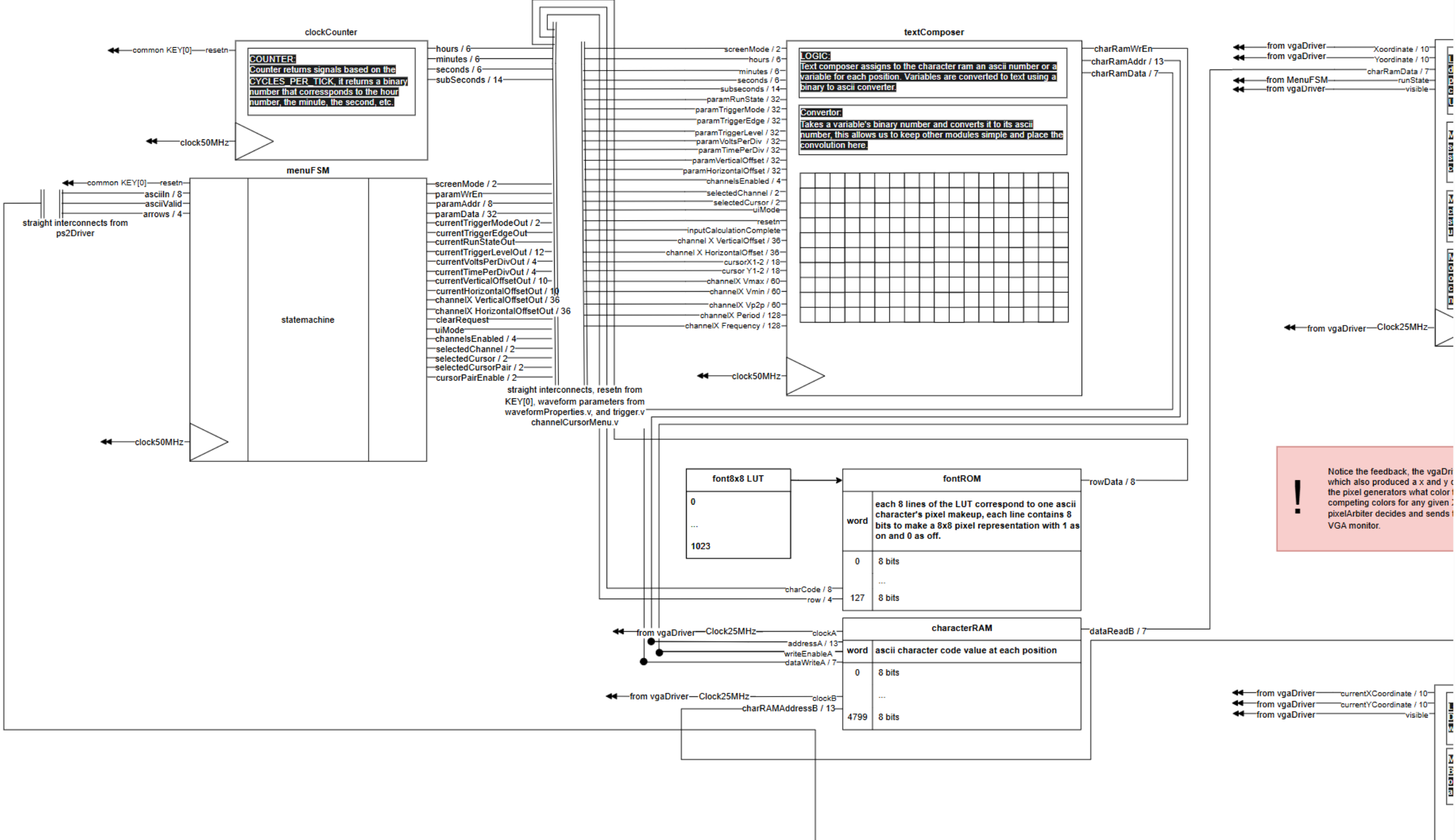
Rahil Harit-Singh and Ayan Ali

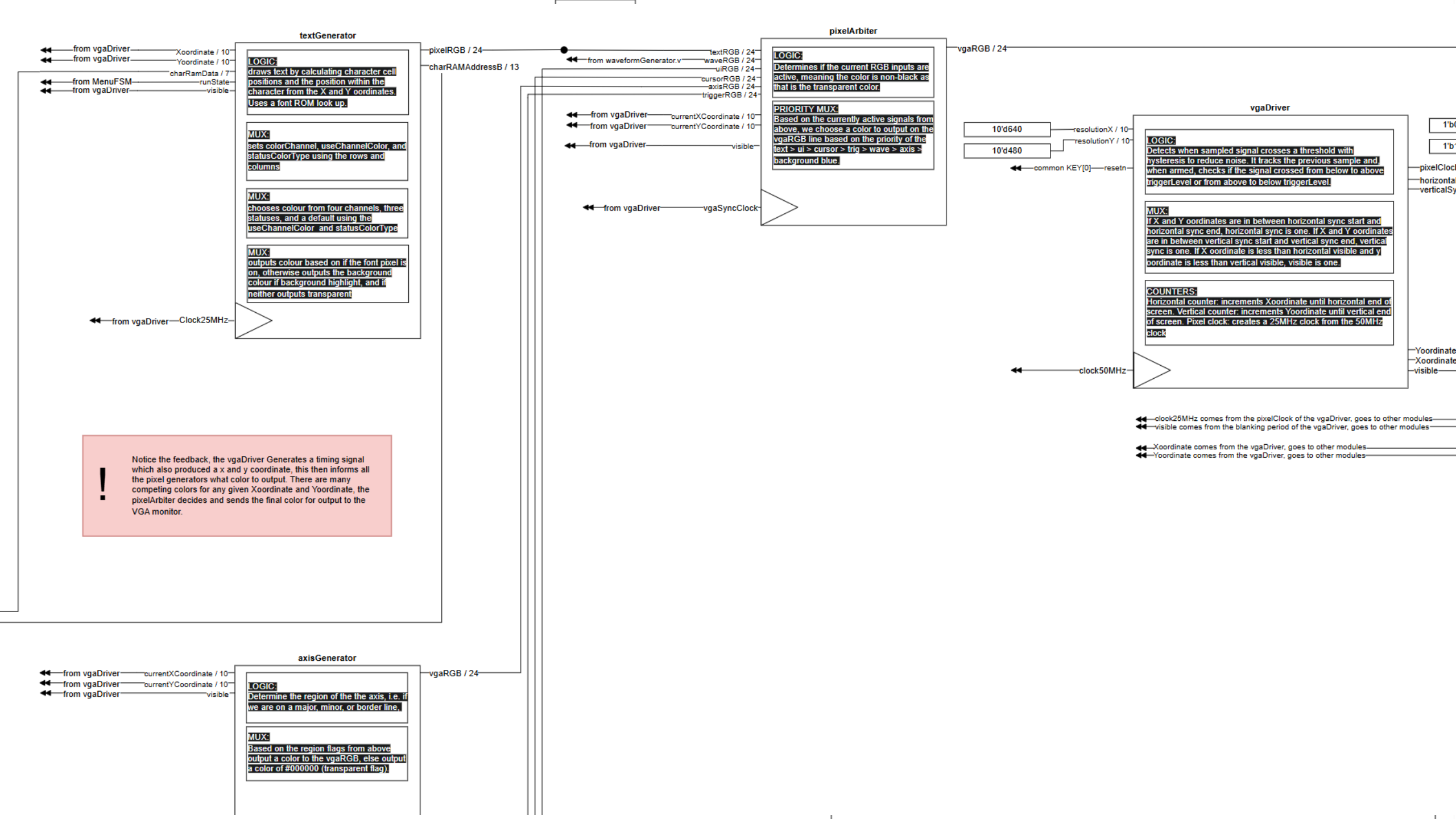


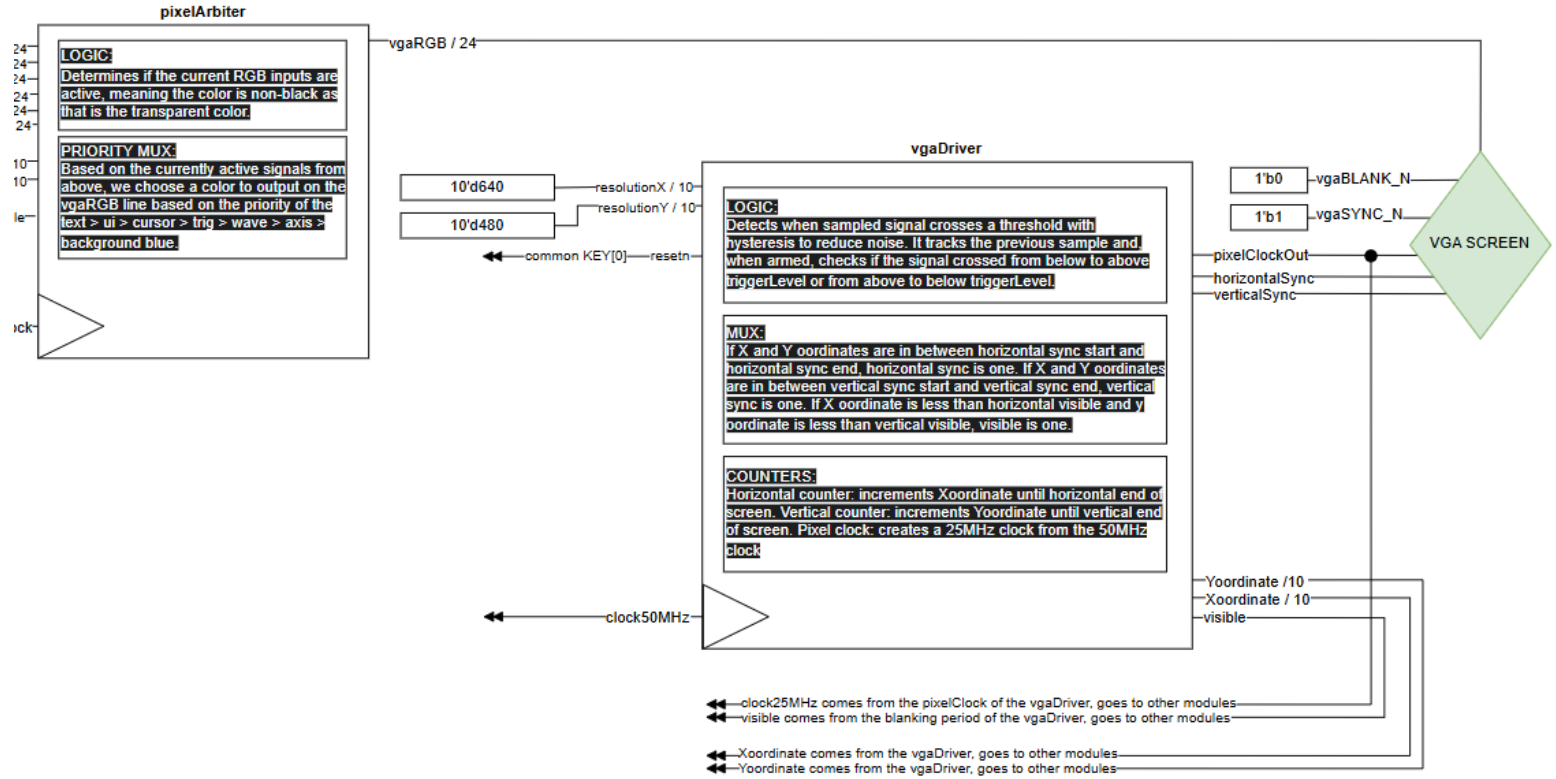
Full Quality: https://drive.google.com/drive/folders/1YEOpY6BVlhRprcqWDSjgAMX5UXb0tUFo?usp=sharing

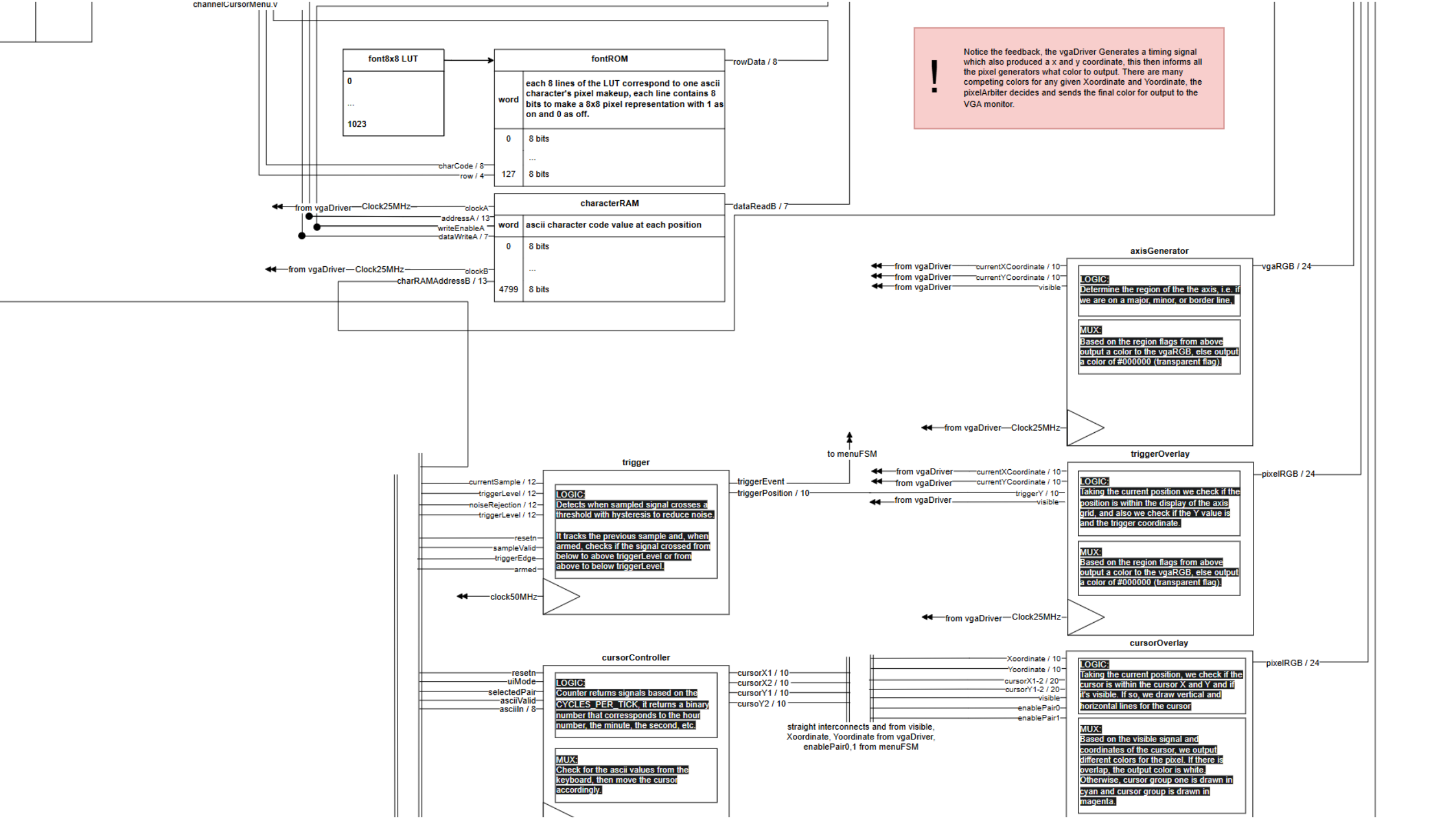


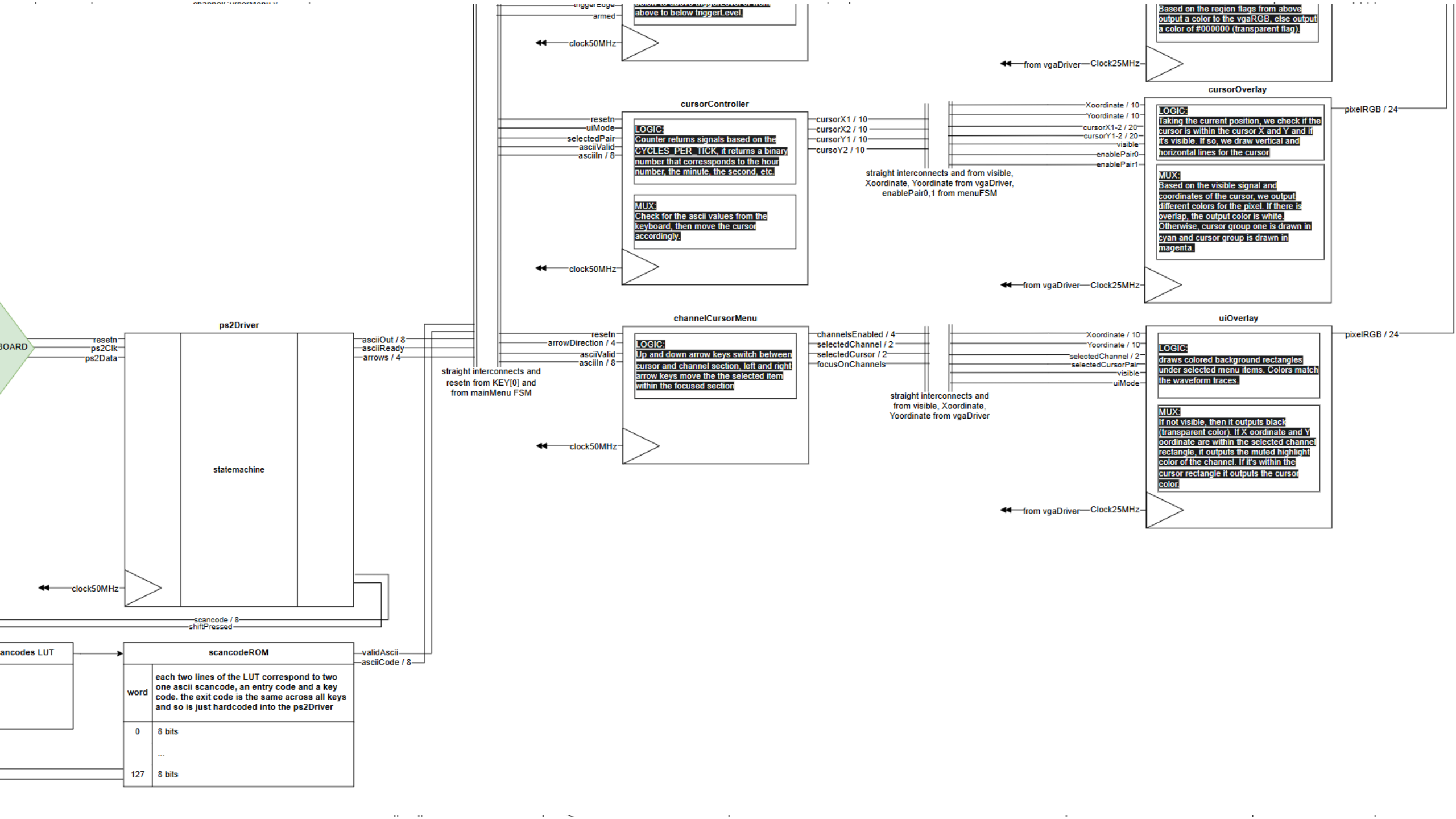
Full Quality: https://drive.google.com/drive/folders/1YEOpY6BVlhRprcqWDSjgAMX5UXb0tUFo?usp=sharing

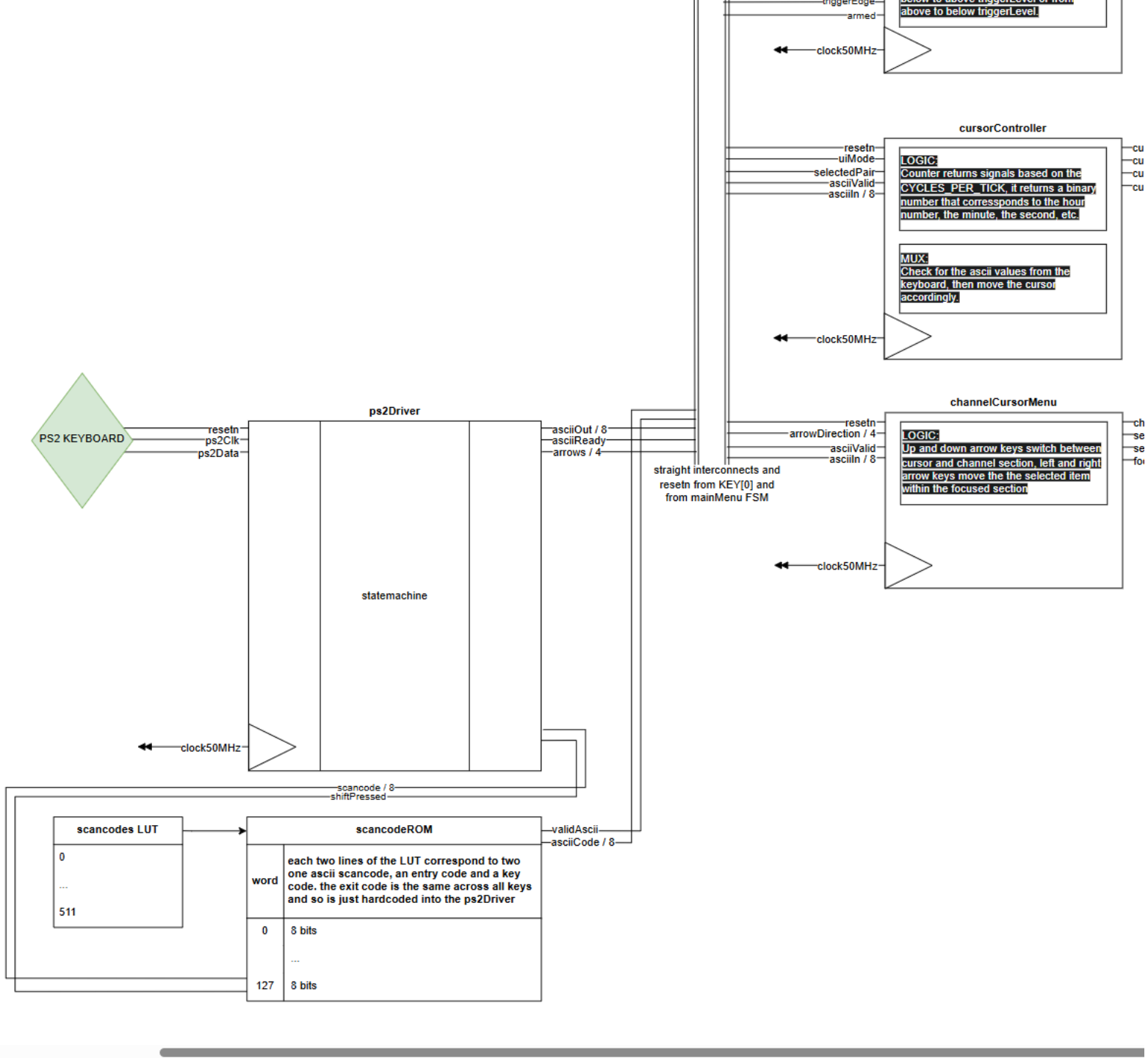

clockCounter
common KEY[0]
resetn
COUNTER:
Counter returns signals based on the CYCLES_PER_TICK, it returns a binary number that corressponds to the hour number, the minute, the second, etc.
clock50MHz
hours / 6
minutes / 6
seconds / 6
subSeconds / 14
menuFSM
common KEY[0]
resetn
asciiIn / 8
asciiValid
arrows / 4
straight interconnects from ps2Driver
statemachine
clock50MHz
screenMode / 2
paramWrEn
paramAddr / 8
paramData / 32
currentTriggerModeOut / 2
currentTriggerEdgeOut
currentRunStateOut
currentTriggerLevelOut / 12
currentVoltsPerDivOut / 4
currentTimePerDivOut / 4
currentVerticalOffsetOut / 10
currentHorizontalOffsetOut / 10
channelX VerticalOffsetOut / 36
channelX HorizontalOffsetOut / 36
clearRequest
uiMode
channelsEnabled / 4
selectedChannel / 2
selectedCursor / 2
selectedCursorPair / 2
cursorPairEnable / 2
straight interconnects, resetn from KEY[0], waveform parameters from waveformProperties.v, and trigger.v
channelCursorMenu.v
textComposer
LOGIC:
Text composer assigns to the character ram an ascii number or a variable for each position. Variables are converted to text using a binary to ascii converter.
Convertor:
Takes a variable's binary number and converts it to its ascii number, this allows us to keep other modules simple and place the convolution here.
clock50MHz
screenMode / 2
hours / 6
minutes / 6
seconds / 6
subseconds / 14
paramRunState / 32
paramTriggerMode / 32
paramTriggerEdge / 32
paramTriggerLevel / 32
paramVoltsPerDiv / 32
paramTimePerDiv / 32
paramVerticalOffset / 32
paramHorizontalOffset / 32
channelsEnabled / 4
selectedChannel / 2
selectedCursor / 2
uiMode
resetn
inputCalculationComplete
channel X VerticalOffset / 36
channel X HorizontalOffset / 36
cursorX1-2 / 18
cursor Y1-2 / 18
channelX Vmax / 60
channelX Vmin / 60
channelX Vp2p / 60
channelX Period / 128
channelX Frequency / 128
charRamWrEn
charRamAddr / 13
charRamData / 7
font8x8 LUT
0
...
1023
fontROM
word
each 8 lines of the LUT correspond to one ascii character's pixel makeup, each line contains 8 bits to make a 8x8 pixel representation with 1 as on and 0 as off.
0
8 bits
...
127
8 bits
rowData / 8
charCode / 8
row / 4
characterRAM
word
ascii character code value at each position
0
8 bits
...
4799
8 bits
from vgaDriver
Clock25MHz
clockA
addressA / 13
writeEnableA
dataWriteA / 7
dataReadB / 7
clockB
charRAMAddressB / 13
from vgaDriver
Xoordinate / 10
Yoordinate / 10
charRamData / 7
from MenuFSM
runState
from vgaDriver
visible
from vgaDriver
Clock25MHz
!
Notice the feedback, the vgaDri
which also produced a x and y c
the pixel generators what color
competing colors for any given
pixelArbiter decides and sends
VGA monitor.
from vgaDriver
currentXCoordinate / 10
from vgaDriver
currentYCoordinate / 10
from vgaDriver
visible

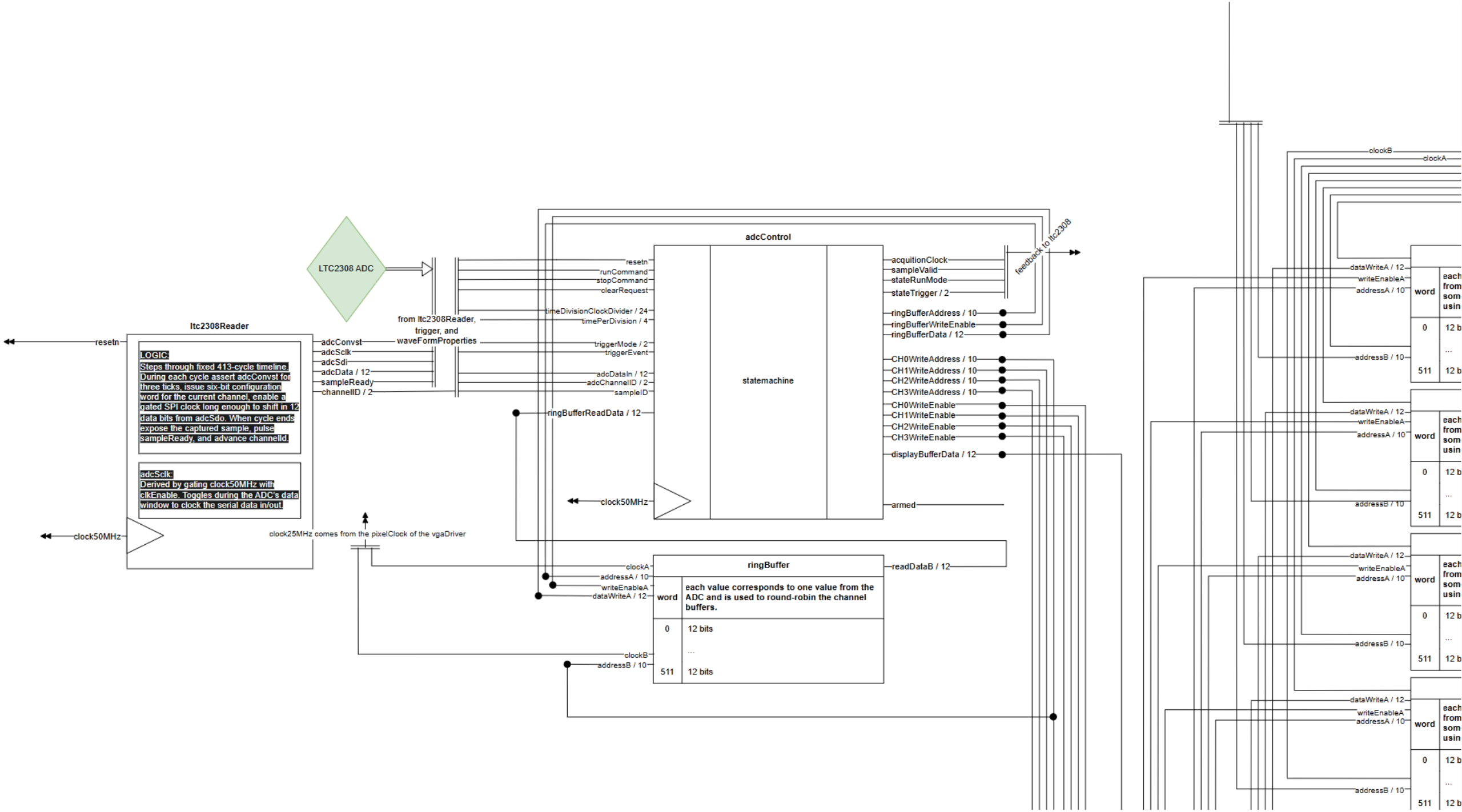

textGenerator
from vgaDriver
Xoordinate / 10
from vgaDriver
Yoordinate / 10
charRamData / 7
from MenuFSM
runState
from vgaDriver
visible
LOGIC:
draws text by calculating character cell positions and the position within the character from the X and Y oordinates. Uses a font ROM look up.
MUX:
sets colorChannel, useChannelColor, and statusColorType using the rows and columns
MUX:
chooses colour from four channels, three statuses, and a default using the useChannelColor and statusColorType
MUX:
outputs colour based on if the font pixel is on, otherwise outputs the background colour if background highlight, and if neither outputs transparent
from vgaDriver
Clock25MHz
pixelRGB / 24
charRAMAddressB / 13
pixelArbiter
textRGB / 24
from waveformGenerator.v
waveRGB / 24
uiRGB / 24
cursorRGB / 24
axisRGB / 24
triggerRGB / 24
from vgaDriver
currentXCoordinate / 10
from vgaDriver
currentYCoordinate / 10
from vgaDriver
visible
from vgaDriver
vgaSyncClock
LOGIC:
Determines if the current RGB inputs are active, meaning the color is non-black as that is the transparent color.
PRIORITY MUX:
Based on the currently active signals from above, we choose a color to output on the vgaRGB line based on the priority of the text > ui > cursor > trig > wave > axis > background blue.
vgaRGB / 24
vgaDriver
10'd640
resolutionX / 10
10'd480
resolutionY / 10
common KEY[0]
resetn
clock50MHz
LOGIC:
Detects when sampled signal crosses a threshold with hysteresis to reduce noise. It tracks the previous sample and, when armed, checks if the signal crossed from below to above triggerLevel or from above to below triggerLevel.
MUX:
If X and Y oordinates are in between horizontal sync start and horizontal sync end, horizontal sync is one. If X and Y oordinates are in between vertical sync start and vertical sync end, vertical sync is one. If X oordinate is less than horizontal visible and y oordinate is less than vertical visible, visible is one.
COUNTERS:
Horizontal counter: increments Xoordinate until horizontal end of screen. Vertical counter: increments Yoordinate until vertical end of screen. Pixel clock: creates a 25MHz clock from the 50MHz clock
pixelClock
horizontal
verticalSy
Yoordinate
Xoordinate
visible
clock25MHz comes from the pixelClock of the vgaDriver, goes to other modules
visible comes from the blanking period of the vgaDriver, goes to other modules
Xoordinate comes from the vgaDriver, goes to other modules
Yoordinate comes from the vgaDriver, goes to other modules
!
Notice the feedback, the vgaDriver Generates a timing signal which also produced a x and y coordinate, this then informs all the pixel generators what color to output. There are many competing colors for any given Xoordinate and Yoordinate, the pixelArbiter decides and sends the final color for output to the VGA monitor.
axisGenerator
from vgaDriver
currentXCoordinate / 10
from vgaDriver
currentYCoordinate / 10
from vgaDriver
visible
vgaRGB / 24
LOGIC:
Determine the region of the the axis, i.e. if we are on a major, minor, or border line.
MUX:
Based on the region flags from above output a color to the vgaRGB, else output a color of #000000 (transparent flag).

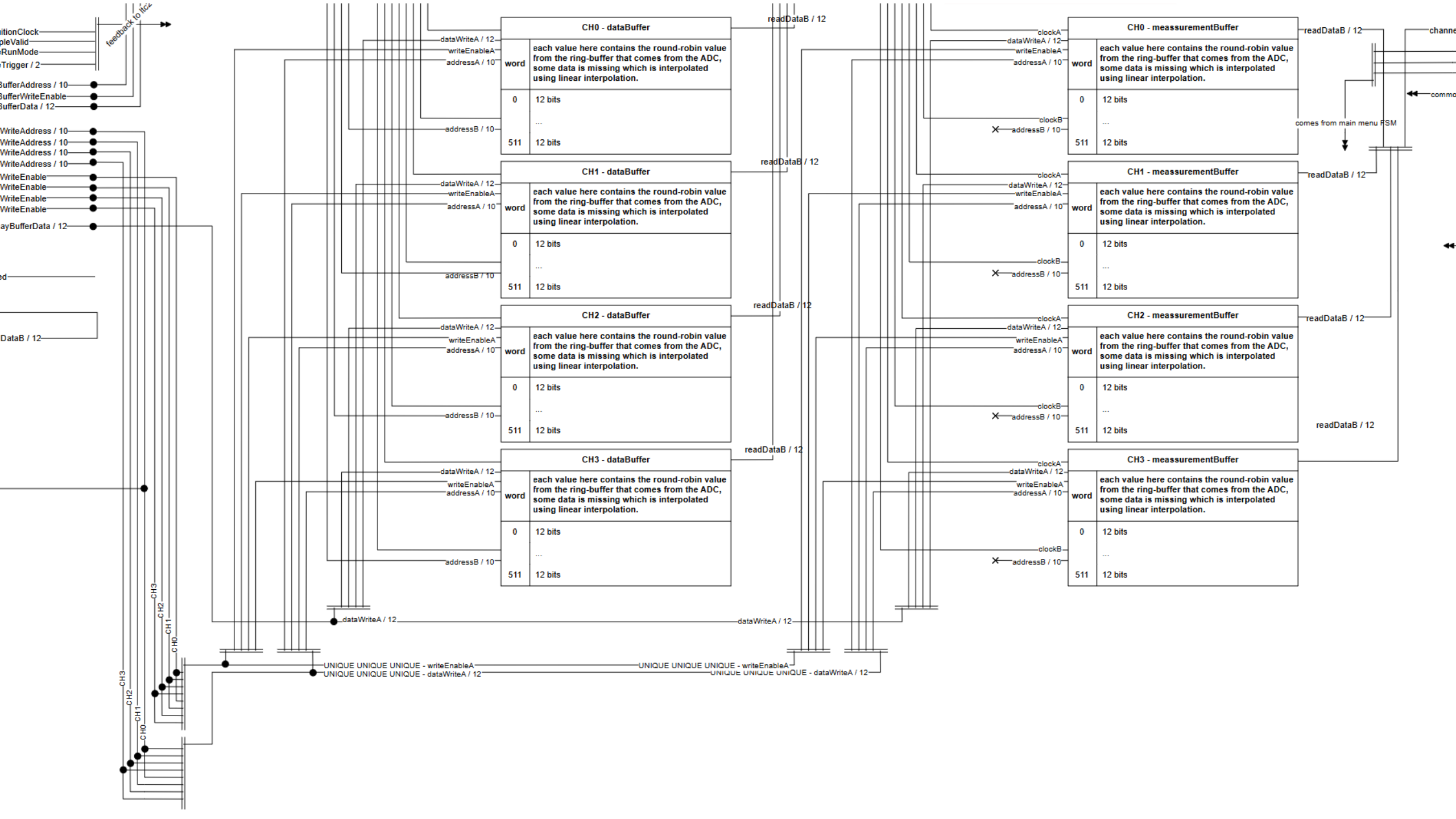

pixelArbiter
LOGIC:
Determines if the current RGB inputs are active, meaning the color is non-black as that is the transparent color.
PRIORITY MUX:
Based on the currently active signals from above, we choose a color to output on the vgaRGB line based on the priority of the text > ui > cursor > trig > wave > axis > background blue.
vgaRGB / 24
vgaDriver
10'd640
10'd480
resolutionX / 10
resolutionY / 10
common KEY[0]
resetn
LOGIC:
Detects when sampled signal crosses a threshold with hysteresis to reduce noise. It tracks the previous sample and, when armed, checks if the signal crossed from below to above triggerLevel or from above to below triggerLevel.
MUX:
If X and Y oordinates are in between horizontal sync start and horizontal sync end, horizontal sync is one. If X and Y oordinates are in between vertical sync start and vertical sync end, vertical sync is one. If X oordinate is less than horizontal visible and y oordinate is less than vertical visible, visible is one.
COUNTERS:
Horizontal counter: increments Xoordinate until horizontal end of screen. Vertical counter: increments Yoordinate until vertical end of screen. Pixel clock: creates a 25MHz clock from the 50MHz clock
clock50MHz
1'b0
vgaBLANK_N
1'b1
vgaSYNC_N
pixelClockOut
horizontalSync
verticalSync
VGA SCREEN
Yoordinate /10
Xoordinate / 10
visible
clock25MHz comes from the pixelClock of the vgaDriver, goes to other modules
visible comes from the blanking period of the vgaDriver, goes to other modules
Xoordinate comes from the vgaDriver, goes to other modules
Yoordinate comes from the vgaDriver, goes to other modules

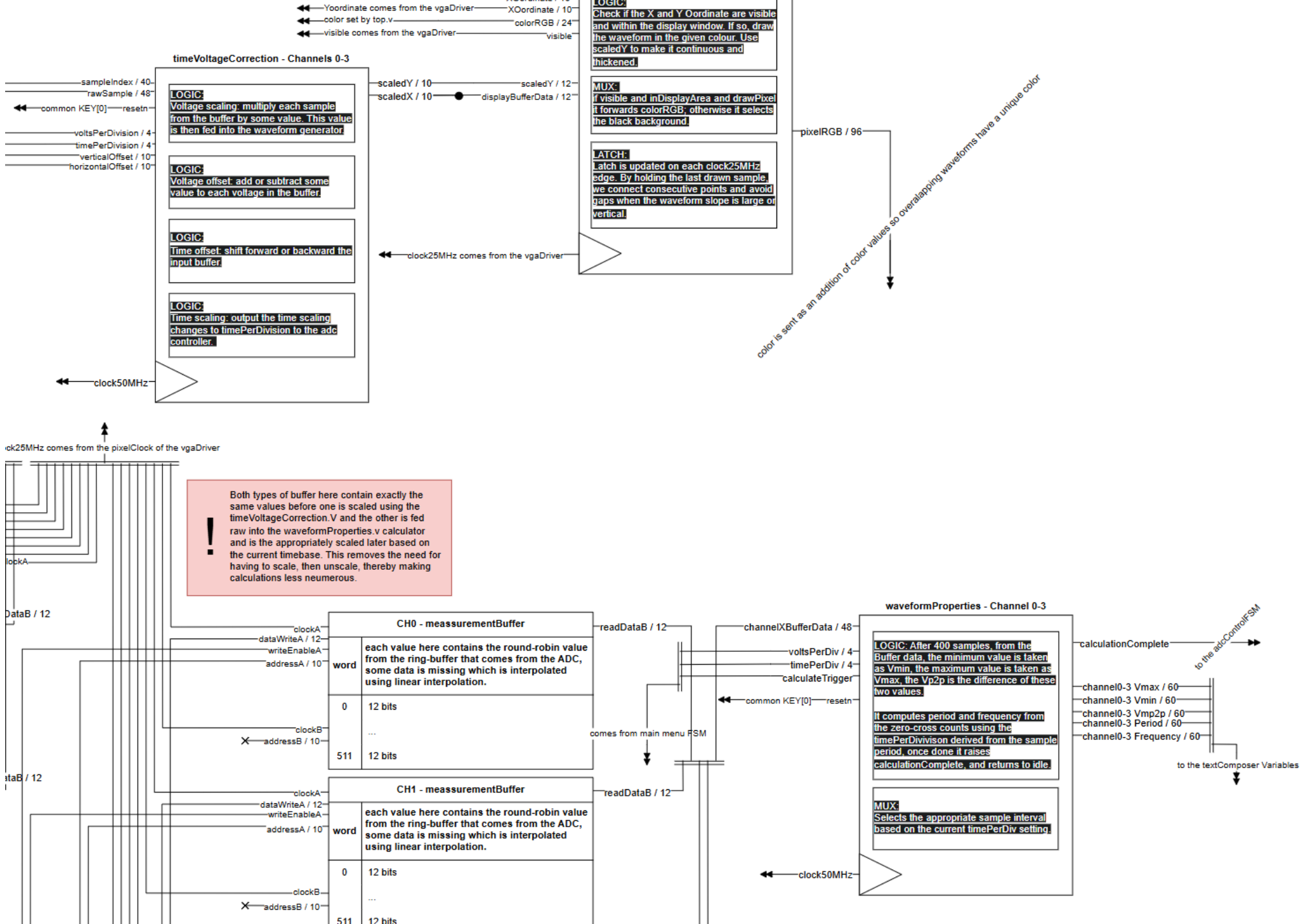

channelCursorMenu.v
font8x8 LUT
0
...
1023
fontROM
word
each 8 lines of the LUT correspond to one ascii character's pixel makeup, each line contains 8 bits to make a 8x8 pixel representation with 1 as on and 0 as off.
0
8 bits
...
127
8 bits
rowData / 8
charCode / 8
row / 4
characterRAM
word
ascii character code value at each position
0
8 bits
...
4799
8 bits
from vgaDriver
Clock25MHz
clockA
addressA / 13
writeEnableA
dataWriteA / 7
dataReadB / 7
clockB
charRAMAddressB / 13
!
Notice the feedback, the vgaDriver Generates a timing signal which also produced a x and y coordinate, this then informs all the pixel generators what color to output. There are many competing colors for any given Xoordinate and Yoordinate, the pixelArbiter decides and sends the final color for output to the VGA monitor.
axisGenerator
currentXCoordinate / 10
currentYCoordinate / 10
visible
LOGIC:
Determine the region of the the axis, i.e. if we are on a major, minor, or border line.
MUX:
Based on the region flags from above output a color to the vgaRGB, else output a color of #000000 (transparent flag).
vgaRGB / 24
trigger
currentSample / 12
triggerLevel / 12
noiseRejection / 12
triggerLevel / 12
resetn
sampleValid
triggerEdge
armed
clock50MHz
LOGIC:
Detects when sampled signal crosses a threshold with hysteresis to reduce noise.
It tracks the previous sample and, when armed, checks if the signal crossed from below to above triggerLevel or from above to below triggerLevel.
triggerEvent
triggerPosition / 10
to menuFSM
triggerOverlay
triggerY / 10
LOGIC:
Taking the current position we check if the position is within the display of the axis grid, and also we check if the Y value is and the trigger coordinate.
MUX:
Based on the region flags from above output a color to the vgaRGB, else output a color of #000000 (transparent flag).
pixelRGB / 24
cursorController
resetn
uiMode
selectedPair
asciiValid
asciiIn / 8
LOGIC:
Counter returns signals based on the CYCLES_PER_TICK, it returns a binary number that corressponds to the hour number, the minute, the second, etc.
MUX:
Check for the ascii values from the keyboard, then move the cursor accordingly.
cursorX1 / 10
cursorX2 / 10
cursorY1 / 10
cursoY2 / 10
straight interconnects and from visible, Xoordinate, Yoordinate from vgaDriver, enablePair0,1 from menuFSM
Xoordinate / 10
Yoordinate / 10
cursorX1-2 / 20
cursorY1-2 / 20
visible
enablePair0
enablePair1
cursorOverlay
LOGIC:
Taking the current position, we check if the cursor is within the cursor X and Y and if it's visible. If so, we draw vertical and horizontal lines for the cursor
MUX:
Based on the visible signal and coordinates of the cursor, we output different colors for the pixel. If there is overlap, the output color is white. Otherwise, cursor group one is drawn in cyan and cursor group is drawn in magenta.


armed
clock50MHz
Based on the region flags from above output a color to the vgaRGB, else output a color of #000000 (transparent flag).
from vgaDriver
Clock25MHz
cursorOverlay
cursorController
resetn
uiMode
selectedPair
asciiValid
asciiIn / 8
LOGIC:
Counter returns signals based on the CYCLES_PER_TICK, it returns a binary number that corressponds to the hour number, the minute, the second, etc.
MUX:
Check for the ascii values from the keyboard, then move the cursor accordingly.
cursorX1 / 10
cursorX2 / 10
cursorY1 / 10
cursoY2 / 10
straight interconnects and from visible, Xoordinate, Yoordinate from vgaDriver, enablePair0,1 from menuFSM
Xoordinate / 10
Yoordinate / 10
cursorX1-2 / 20
cursorY1-2 / 20
visible
enablePair0
enablePair1
LOGIC:
Taking the current position, we check if the cursor is within the cursor X and Y and if it's visible. If so, we draw vertical and horizontal lines for the cursor
MUX:
Based on the visible signal and coordinates of the cursor, we output different colors for the pixel. If there is overlap, the output color is white. Otherwise, cursor group one is drawn in cyan and cursor group is drawn in magenta.
pixelRGB / 24
ps2Driver
resetn
ps2Clk
ps2Data
asciiOut / 8
asciiReady
arrows / 4
statemachine
straight interconnects and resetn from KEY[0] and from mainMenu FSM
channelCursorMenu
arrowDirection / 4
LOGIC:
Up and down arrow keys switch between cursor and channel section, left and right arrow keys move the the selected item within the focused section
channelsEnabled / 4
selectedChannel / 2
selectedCursor / 2
focusOnChannels
straight interconnects and from visible, Xoordinate, Yoordinate from vgaDriver
selectedChannel / 2
selectedCursorPair
uiMode
uiOverlay
LOGIC:
draws colored background rectangles under selected menu items. Colors match the waveform traces.
MUX:
If not visible, then it outputs black (transparent color). If X oordinate and Y oordinate are within the selected channel rectangle, it outputs the muted highlight color of the channel. If it's within the cursor rectangle it outputs the cursor color.
scancode / 8
shiftPressed
ancodes LUT
scancodeROM
validAscii
asciiCode / 8
word
each two lines of the LUT correspond to two one ascii scancode, an entry code and a key code. the exit code is the same across all keys and so is just hardcoded into the ps2Driver
0
8 bits
...
127
8 bits


triggerEdge
armed
above to below triggerLevel.
clock50MHz
cursorController
resetn
uiMode
selectedPair
asciiValid
asciiIn / 8
LOGIC:
Counter returns signals based on the CYCLES_PER_TICK, it returns a binary number that corressponds to the hour number, the minute, the second, etc.
MUX:
Check for the ascii values from the keyboard, then move the cursor accordingly.
clock50MHz
channelCursorMenu
resetn
arrowDirection / 4
asciiValid
asciiIn / 8
LOGIC:
Up and down arrow keys switch between cursor and channel section, left and right arrow keys move the the selected item within the focused section
clock50MHz
PS2 KEYBOARD
ps2Driver
resetn
ps2Clk
ps2Data
asciiOut / 8
asciiReady
arrows / 4
statemachine
clock50MHz
straight interconnects and resetn from KEY[0] and from mainMenu FSM
scancode / 8
shiftPressed
scancodes LUT
0
...
511
scancodeROM
validAscii
asciiCode / 8
word
each two lines of the LUT correspond to two one ascii scancode, an entry code and a key code. the exit code is the same across all keys and so is just hardcoded into the ps2Driver
0
8 bits
...
127
8 bits


LTC2308 ADC
from ltc2308Reader, trigger, and waveFormProperties
ltc2308Reader
resetn
LOGIC:
Steps through fixed 413-cycle timeline. During each cycle assert adcConvst for three ticks, issue six-bit configuration word for the current channel, enable a gated SPI clock long enough to shift in 12 data bits from adcSdo. When cycle ends expose the captured sample, pulse sampleReady, and advance channelId.
adcSclk:
Derived by gating clock50MHz with clkEnable. Toggles during the ADC's data window to clock the serial data in/out.
clock50MHz
clock25MHz comes from the pixelClock of the vgaDriver
adcConvst
adcSclk
adcSdi
adcData / 12
sampleReady
channelID / 2
adcControl
resetn
runCommand
stopCommand
clearRequest
timeDivisionClockDivider / 24
timePerDivision / 4
triggerMode / 2
triggerEvent
adcDataIn / 12
adcChannelID / 2
sampleID
ringBufferReadData / 12
clock50MHz
statemachine
acquitionClock
sampleValid
stateRunMode
stateTrigger / 2
feedback to ltc2308
ringBufferAddress / 10
ringBufferWriteEnable
ringBufferData / 12
CH0WriteAddress / 10
CH1WriteAddress / 10
CH2WriteAddress / 10
CH3WriteAddress / 10
CH0WriteEnable
CH1WriteEnable
CH2WriteEnable
CH3WriteEnable
displayBufferData / 12
armed
clockA
addressA / 10
writeEnableA
dataWriteA / 12
ringBuffer
word
each value corresponds to one value from the ADC and is used to round-robin the channel buffers.
0
12 bits
...
511
12 bits
readDataB / 12
clockB
addressB / 10
clockB
clockA
dataWriteA / 12
writeEnableA
addressA / 10
word
addressB / 10
0
511
12 b


CH0 - dataBuffer
CH1 - dataBuffer
CH2 - dataBuffer
CH3 - dataBuffer
CH0 - meassurementBuffer
CH1 - meassurementBuffer
CH2 - meassurementBuffer
CH3 - meassurementBuffer
word
each value here contains the round-robin value from the ring-buffer that comes from the ADC, some data is missing which is interpolated using linear interpolation.
0
12 bits
511
12 bits
dataWriteA / 12
writeEnableA
addressA / 10
addressB / 10
readDataB / 12
clockA
clockB
comes from main menu FSM
feedback to ltc
dataWriteA / 12
UNIQUE UNIQUE UNIQUE - writeEnableA
UNIQUE UNIQUE UNIQUE - dataWriteA / 12
CH3
CH2
CH1
CH0

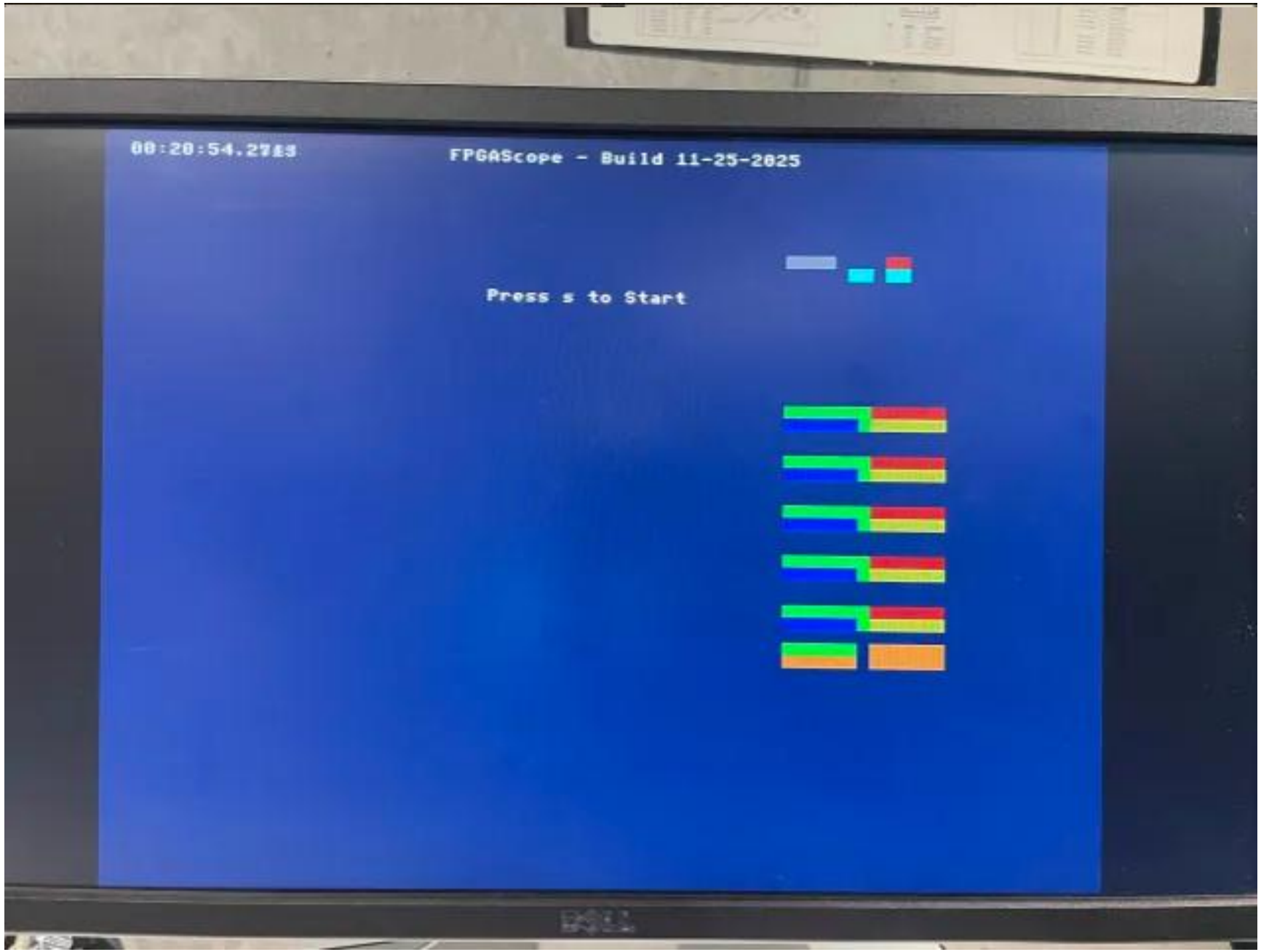

Yoordinate comes from the vgaDriver
XOordinate / 10
color set by top.v
colorRGB / 24
visible comes from the vgaDriver
visible
timeVoltageCorrection - Channels 0-3
sampleIndex / 40
rawSample / 48
common KEY[0]
resetn
voltsPerDivision / 4
timePerDivision / 4
verticalOffset / 10
horizontalOffset / 10
LOGIC:
Voltage scaling: multiply each sample from the buffer by some value. This value is then fed into the waveform generator.
LOGIC:
Voltage offset: add or subtract some value to each voltage in the buffer.
LOGIC:
Time offset: shift forward or backward the input buffer.
LOGIC:
Time scaling: output the time scaling changes to timePerDivision to the adc controller.
clock50MHz
scaledY / 10
scaledX / 10
scaledY / 12
displayBufferData / 12
clock25MHz comes from the vgaDriver
LOGIC:
Check if the X and Y Oordinate are visible and within the display window. If so, draw the waveform in the given colour. Use scaledY to make it continuous and thickened.
MUX:
if visible and inDisplayArea and drawPixel it forwards colorRGB, otherwise it selects the black background.
LATCH:
Latch is updated on each clock25MHz edge. By holding the last drawn sample, we connect consecutive points and avoid gaps when the waveform slope is large or vertical.
pixelRGB / 96
color is sent as an addition of color values so overalapping waveforms have a unique color
ck25MHz comes from the pixelClock of the vgaDriver
lockA
DataB / 12
taB / 12
Both types of buffer here contain exactly the same values before one is scaled using the timeVoltageCorrection.V and the other is fed raw into the waveformProperties.v calculator and is the appropriately scaled later based on the current timebase. This removes the need for having to scale, then unscale, thereby making calculations less neumerous.
CH0 - meassurementBuffer
clockA
dataWriteA / 12
writeEnableA
addressA / 10
word
each value here contains the round-robin value from the ring-buffer that comes from the ADC, some data is missing which is interpolated using linear interpolation.
0
12 bits
...
511
12 bits
clockB
addressB / 10
readDataB / 12
CH1 - meassurementBuffer
clockA
dataWriteA / 12
writeEnableA
addressA / 10
word
each value here contains the round-robin value from the ring-buffer that comes from the ADC, some data is missing which is interpolated using linear interpolation.
0
12 bits
...
511
12 bits
clockB
addressB / 10
readDataB / 12
comes from main menu FSM
waveformProperties - Channel 0-3
channelXBufferData / 48
voltsPerDiv / 4
timePerDiv / 4
calculateTrigger
common KEY[0]
resetn
LOGIC: After 400 samples, from the Buffer data, the minimum value is taken as Vmin, the maximum value is taken as Vmax, the Vp2p is the difference of these two values.
It computes period and frequency from the zero-cross counts using the timePerDivivison derived from the sample period, once done it raises calculationComplete, and returns to idle.
MUX:
Selects the appropriate sample interval based on the current timePerDiv setting.
clock50MHz
calculationComplete
to the adcControlFSM
channel0-3 Vmax / 60
channel0-3 Vmin / 60
channel0-3 Vmp2p / 60
channel0-3 Period / 60
channel0-3 Frequency / 60
to the textComposer Variables

FPGAScope

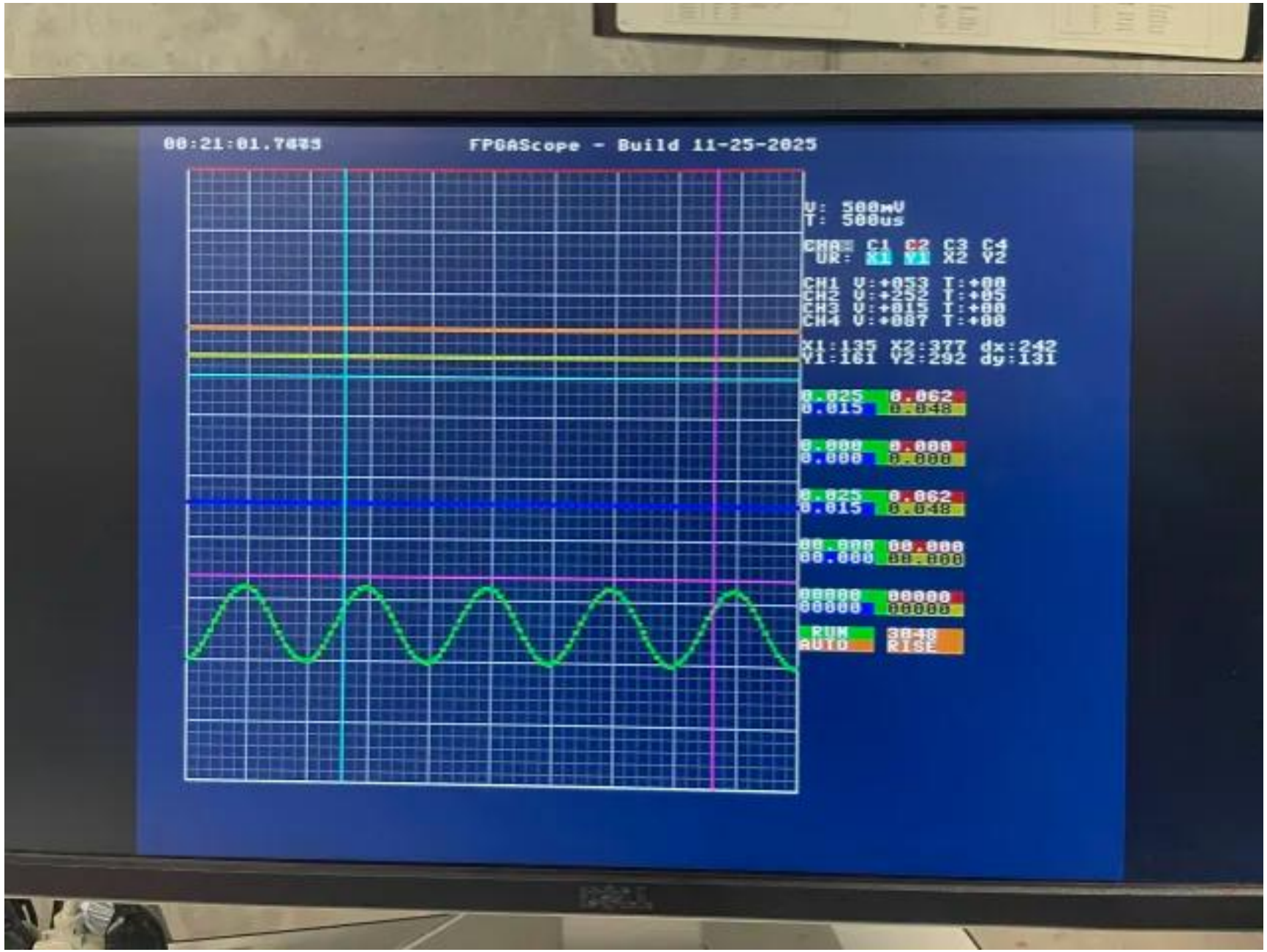
ECE241 Final Project

Project in Action

Rahil Harit-Singh and Ayan Ali


00:20:54.2783
FPGAScope - Build 11-25-2025
Press s to Start


00:21:05.6113
FPGAScope - Build 11-25-2025
KEYBOARD CONTROLS
s - start oscilloscope
h - show help
q - quit back
r - run/stop
t - trigger mode c
e - trigger edge c
w - level up
d - level down
v/V - volts/div
a/z - time/div

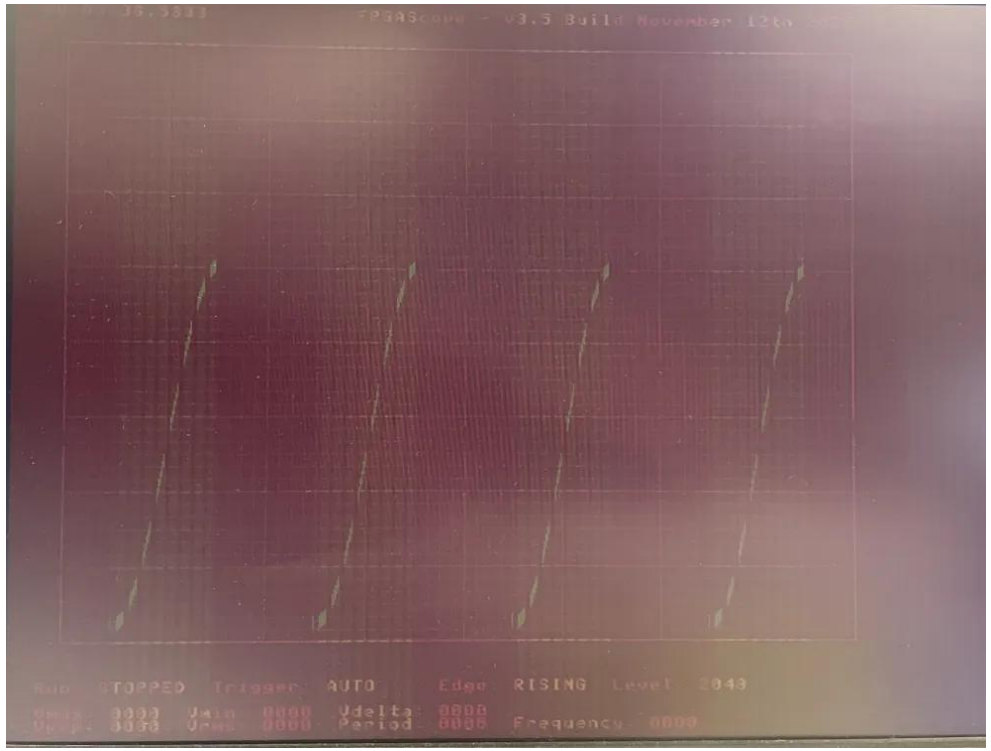

00:21:01.7485
FPGAScope - Build 11-25-2025
V: 500mV
T: 500us
CHA= C1 C2 C3 C4
UR: X1 Y1 X2 Y2
CH1 V:+053 T:+00
CH2 V:+252 T:+05
CH3 V:+015 T:+00
CH4 V:+007 T:+00
X1:135 X2:377 dx:242
Y1:161 Y2:292 dy:131
0.025 0.062
0.015 0.048
0.000 0.000
0.000 0.000
0.025 0.062
0.015 0.048
00.000 00.000
00.000 00.000
00000 00000
00000 00000
RUN 3048
AUTO RISE

FPGAScope

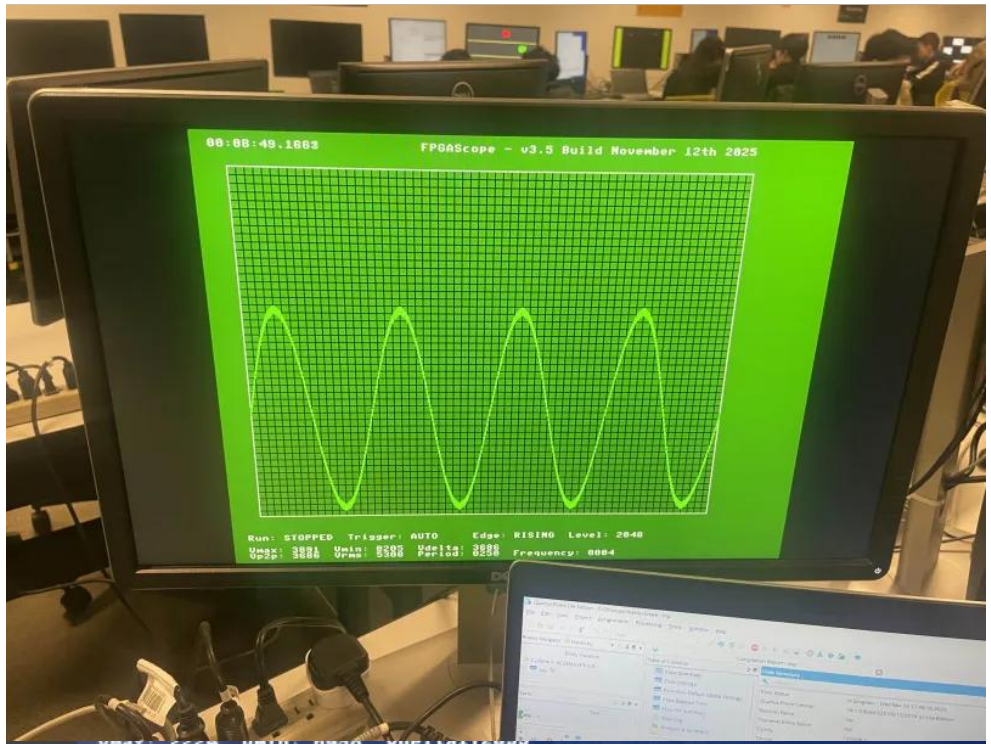
ECE241 Final Project

Bug Fixes

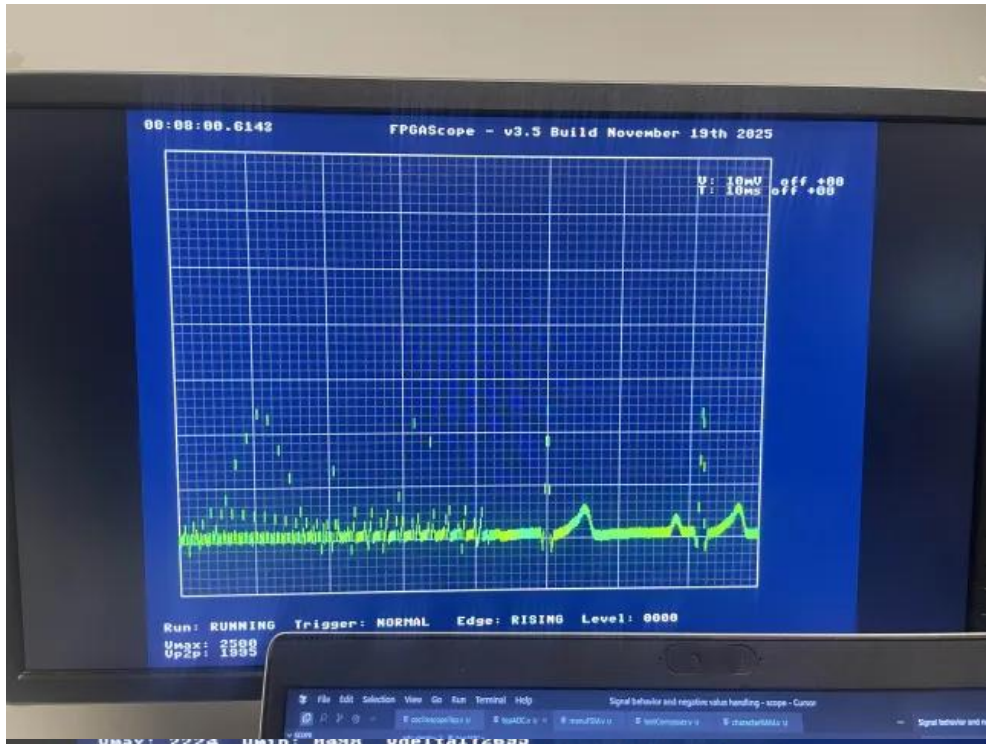
Rahil Harit-Singh and Ayan Ali



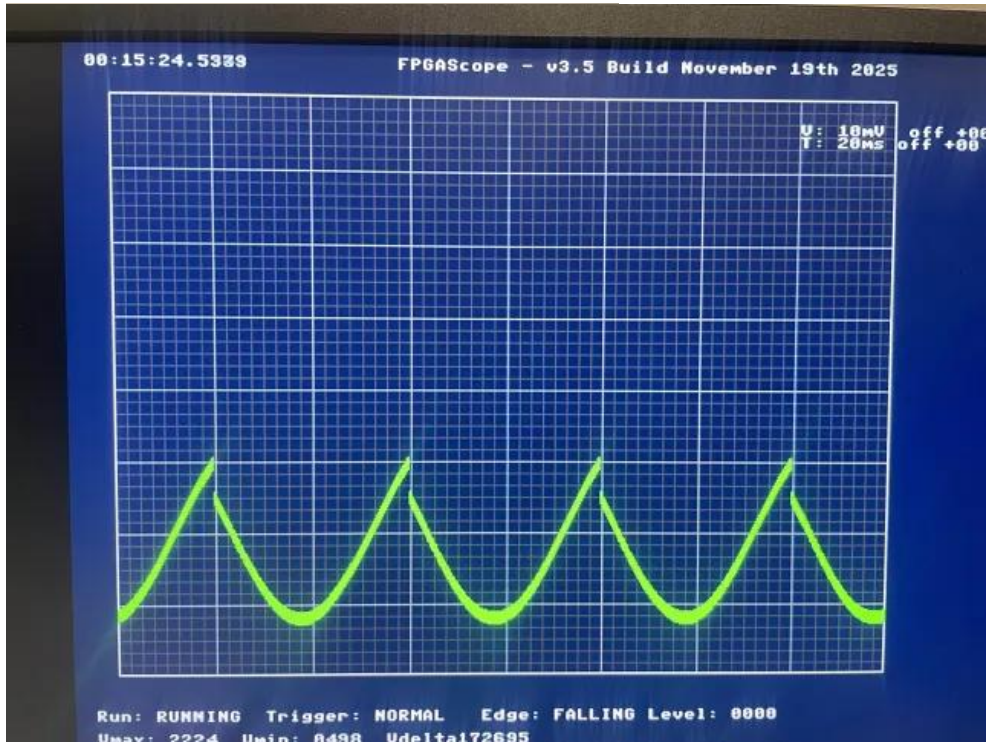
This display bug happened as my code handles only 4-bit color in an effort to reduce displayBuffer.v size. The vga screen expects a 8-bit color for each RGB. This fix was to add extra bits to the end of the color value when passing to VGA.



This display bug happened when the green and blue connections wires were flipped due to a typo.



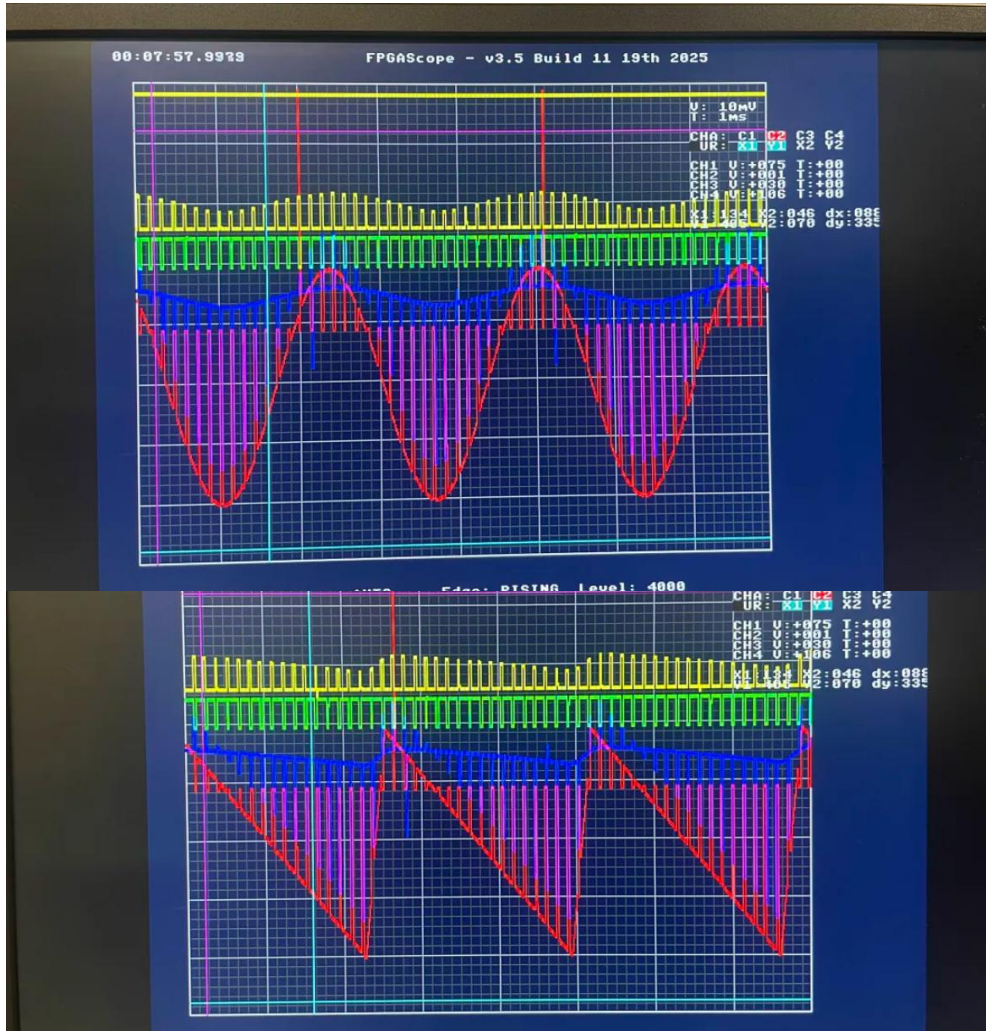
This bug has happened when experimenting with a dynamic buffer size, the issue was that for every increase of the time-scale less data would be able to be displayed per pixel slice of the screen and so it would take smaller and smaller portions of the screen from the left hand side, hence you can see the most zoomed in section on the left.



In an effort to fix this, we tried to repeat the data across the screen to span the full area, but this would simply result in repetitions of the same waveform with cutoffs.

We resolved the final fix would be to actually increase the number of samples taken, i.e. increase the size of the sampling ring buffer and then sample as much or as little as needed.

The issue here is we run into an acquisition time issue, this can be solved by having a dynamic sample rate to only store as much as needed in the buffer.



Here we see that two channels seem to blend into one channel and the other channels have lots of noise. Taking each problem sequentially:

1. The spikes comes from the fact the since we were round-robbining our samples from the ADC we left some holes in our buffers, this was solved using linear interpolation.
2. The two channels as one comes down to timing issues some data bleeding from CH1 into CH2, this was solved by making sure our ltc2308Reader.v file was using the correct clock cycles.
3. The noise is a result of electromagnetic noise between the pins in the ADC, lowering the amplitude solves this issue. In the future we would aim to have an external multi-channel ADC so we don’t have to share polling between channels.

FPGAScope

ECE241 Final Project

Project Demo

Rahil Harit-Singh and Ayan Ali

FPGAScope

ECE241 Final Project

Future Work

Rahil Harit-Singh and Ayan Ali

new ADC

textComposer.v

bugs

KSPS

interpolation

aliasing

1. One quick way of improving the scope and in turn reducing complexity would be to use the GPIO pins and have bipolar-multi-channel (or multiple) ADCs so that we are able to remove our stop-gap to make multi-channel work. We would also be able to have a larger voltage range and be able to display negative values such that we don't need a voltage offset.
2. Some other fixes would be to rewrite our textComposer.v file to use functions, use systemVerilog, or to call from a custom LUT that is generated using a python script rather than having to edit a 2300+ line file.
3. With these two additions, we would be able to get rid of a huge amount of complexity thereby reducing our bugs significantly and allowing us to solve all remaining bugs and flesh out all remaining features.
4. Increasing our sampling rate would allow us to sample higher frequencies without loosing too much data.
5. Using a sinc interpolation would better than linear interpolation, making our zoomed-in waveforms at high frequency cleaner.
6. Increasing screen resolution or other buffer size-related issues would get rid of aliasing and allow us to see zoomed out waves in full detail.

FPGAScope

ECE241 Final Project

Thank you!

Rahil Harit-Singh and Ayan Ali