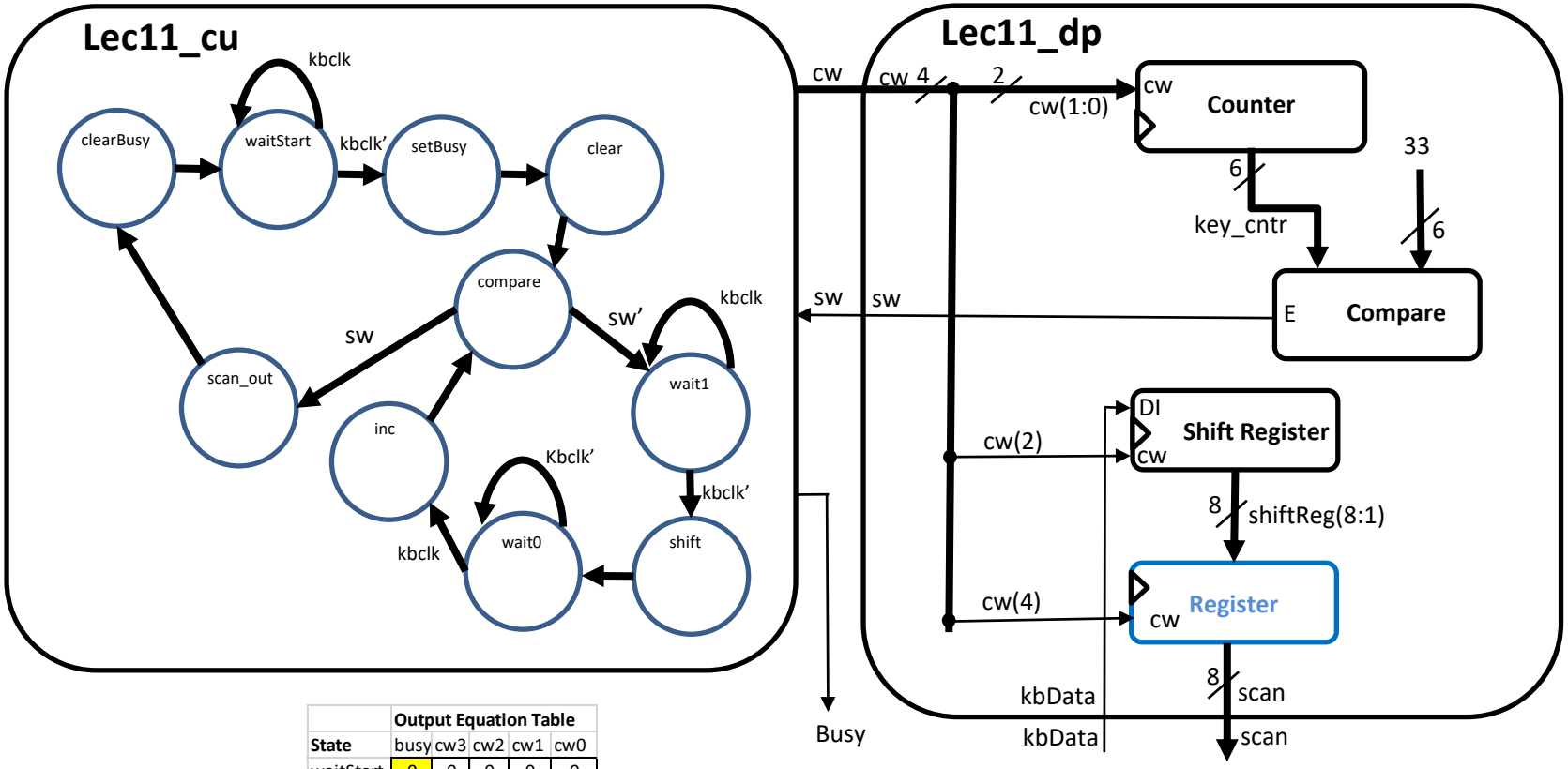
UNITED STATES
AIR FORCE
ACADEMY

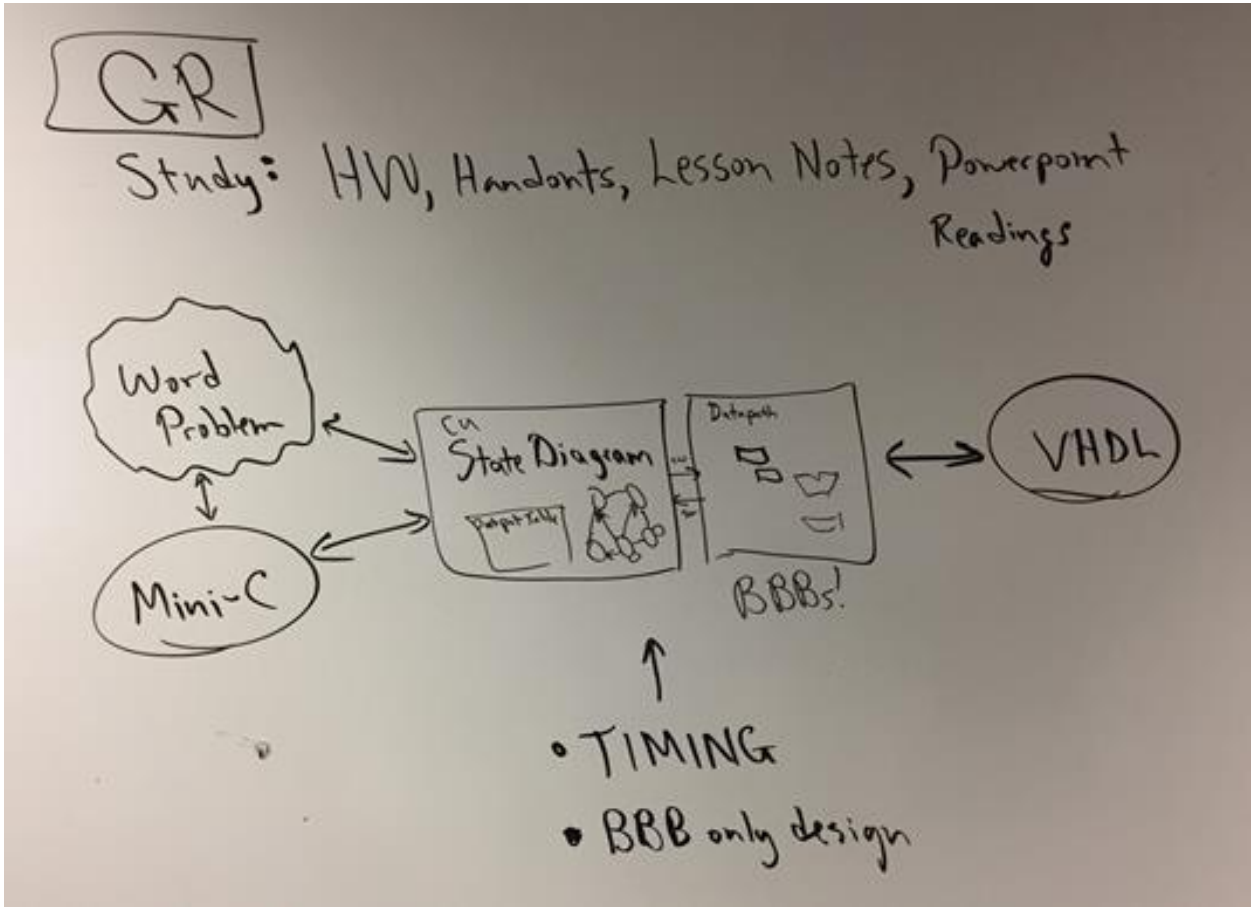
# ECE 383 – Embedded Computer Systems II
## Lecture 13 – Lab 2 – Data Acquisition, Storage and Display

# Feedback on HW#8 (GR?)




| | Output Equation Table | | | | |
|---|---|---|---|---|---|
| **State** | busy | cw3 | cw2 | cw1 | cw0 |
| waitStart | 0 | 0 | 0 | 0 | 0 |
| setBusy | 1 | 0 | 0 | 0 | 0 |
| clear | 1 | 0 | 0 | 1 | 1 |
| compare | 1 | 0 | 0 | 0 | 0 |
| wait1 | 1 | 0 | 0 | 0 | 0 |
| inc | 1 | 0 | 0 | 0 | 1 |
| wait0 | 1 | 0 | 0 | 0 | 0 |
| scan_out | 0 | 1 | 0 | 0 | 0 |
| shift | 1 | 0 | 1 | 0 | 0 |

# GR?

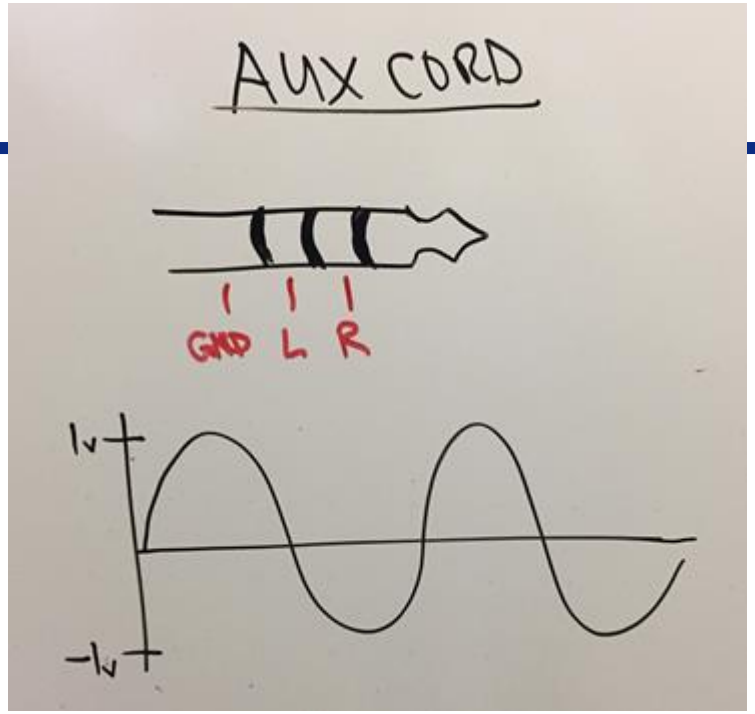

GR
Study: HW, Handouts, Lesson Notes, Powerpoint
Readings
Word Problem
Mini-C
CU
State Diagram
Datapath
VHDL
BBBs!
• TIMING
• BBB only design

UNITED STATES AIR FORCE ACADEMY

# Lesson Outline

- ~~Tips and Tricks~~
- Lab 2 – Data Acquisition, Storage and Display

extra lesson this year




Audio ADC

Sampling at 48KHz

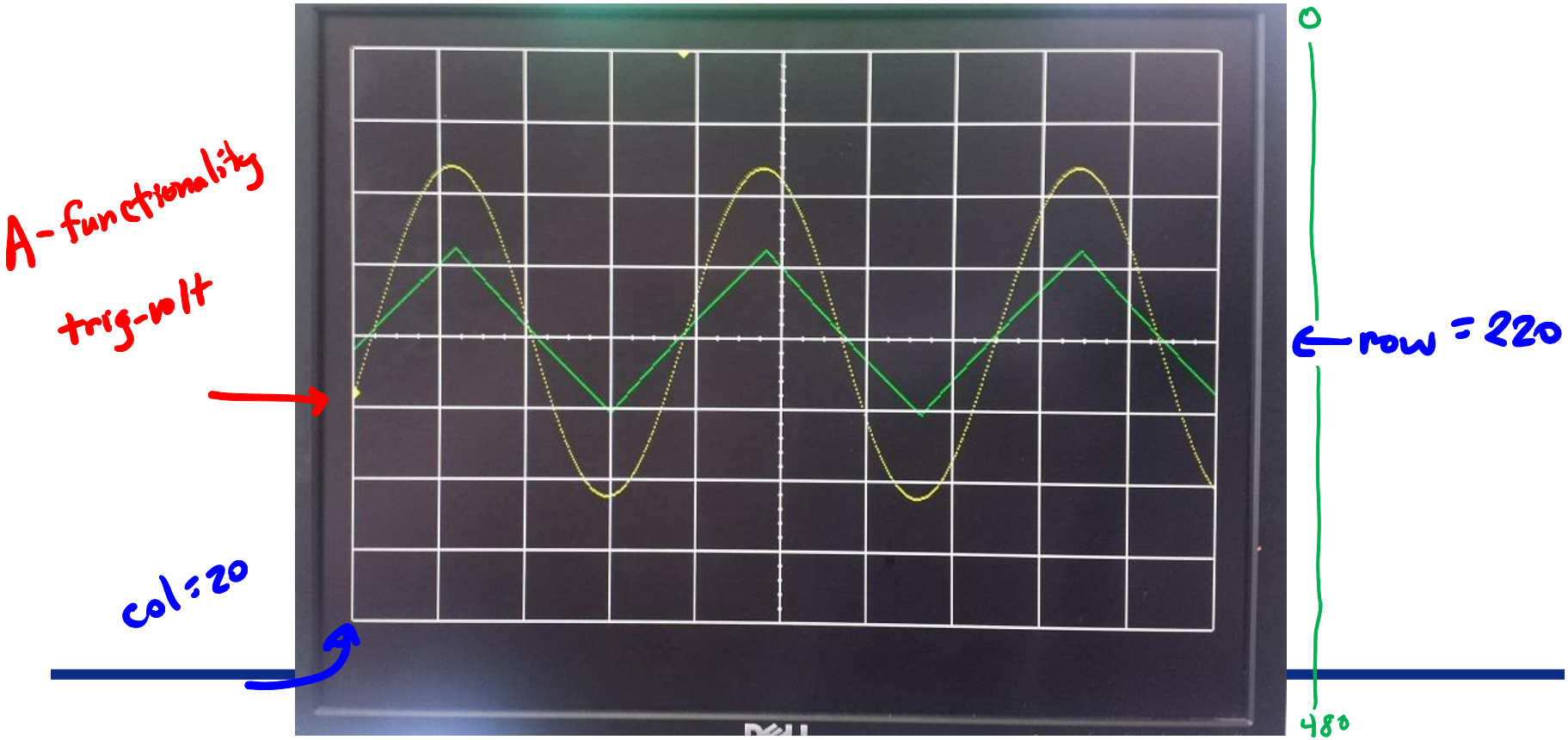
or 24kHz?

# Lab 2 – Data Acquisition, Storage and Display

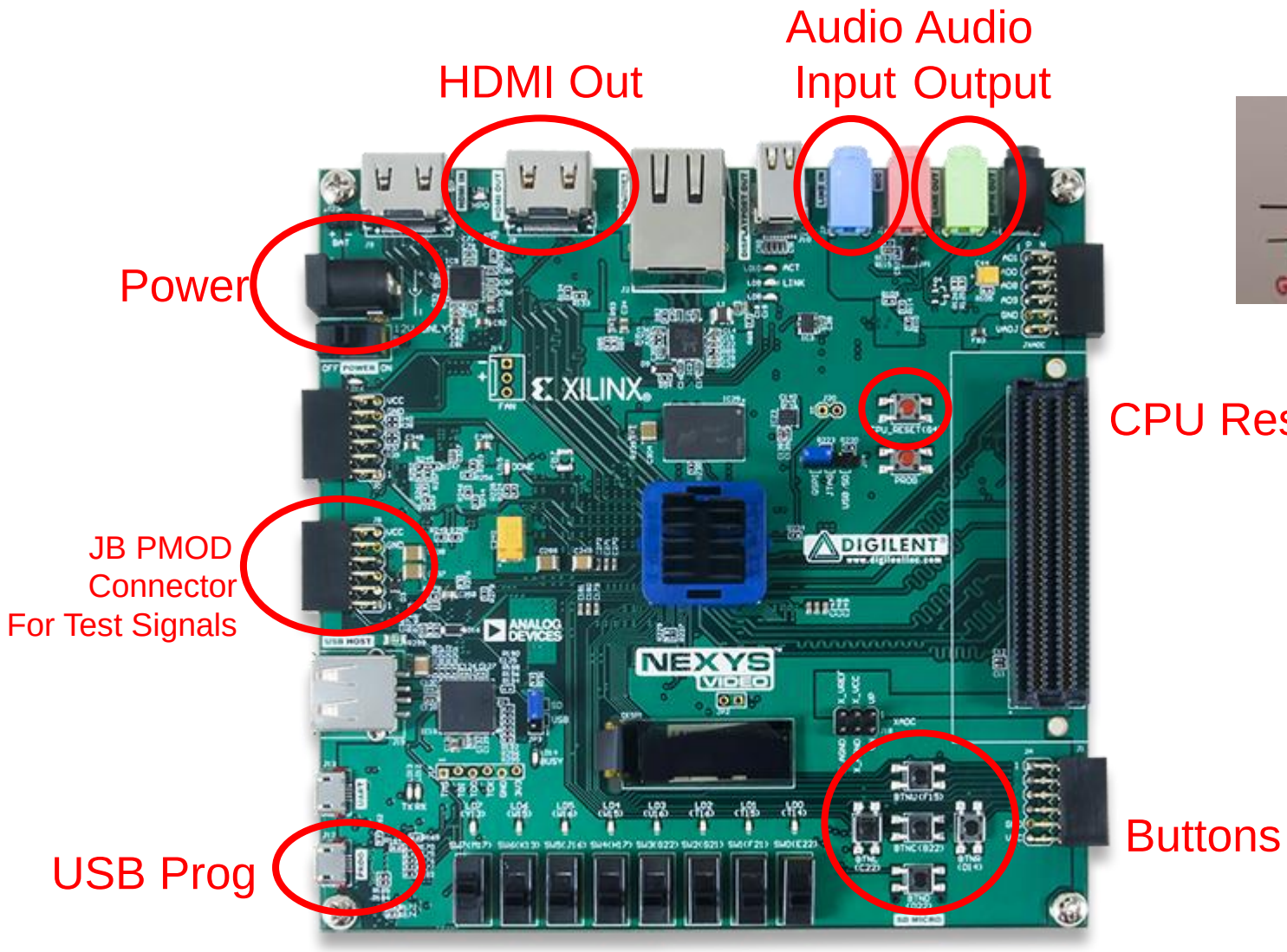
UNITED STATES AIR FORCE ACADEMY

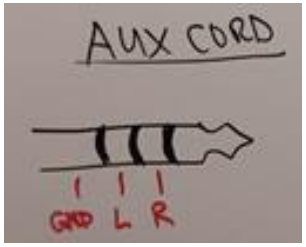
# Lab 2 – Lab Overview

- Lab Overview - Integrate the video display controller developed in Lab 1 with the audio codec on the Nexys Video board to build a basic 2-channel oscilloscope.



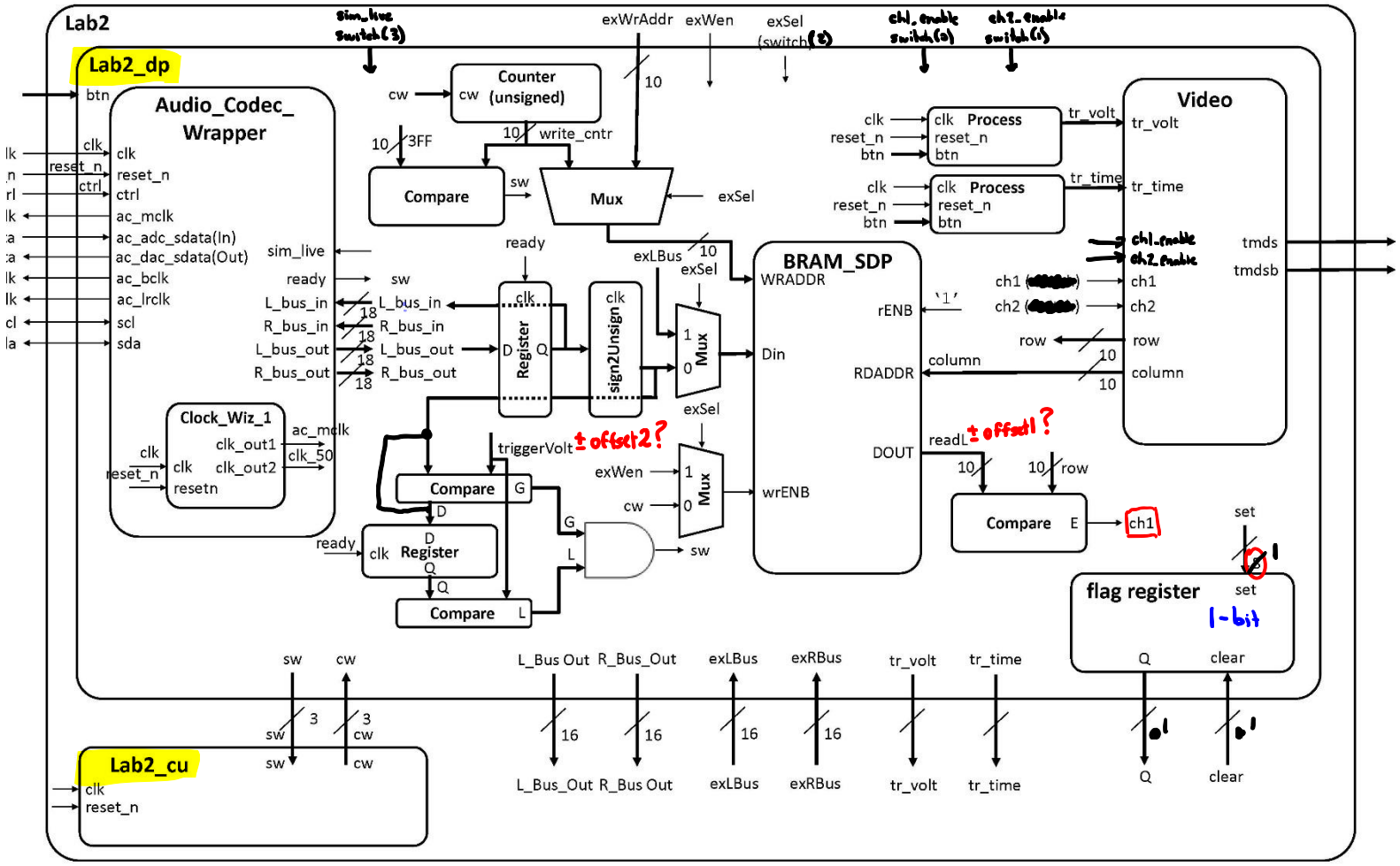
# Lab 2 – Connections



UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Architecture



# Lab 2 – ADAU1761 SigmaDSP Audio Codec

- Analog Devices ADAU1761 SigmaDSP Audio Codec.
  - http://www.analog.com/media/en/technical-documentation/data-sheets/ADAU1761.pdf
- 1. Loop back L_bus_out and R_bus_out and listen on the HP_OUT Jack

```
process (clk)
   begin
      if (rising_edge(clk)) then
         if reset_n = '0' then
                  L_bus_in <= (others => '0');
                  R_bus_in <= (others => '0');
         elsif(ready = '1') then
                  L_bus_in <= L_bus_out;
                  R_bus_in <= R_bus_out;
         end if;
      end if;
end process;
```

Gate check 3 ? 2 ?

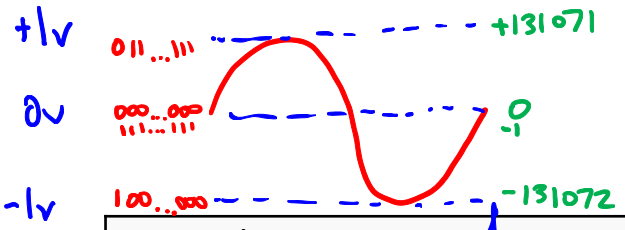
# Lab 2 – ADAU1761 SigmaDSP Audio Codec

UNITED STATES AIR FORCE ACADEMY

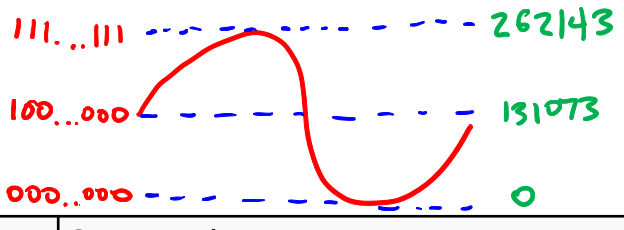
Not Casting!!!

- 2. Convert L_bus_out signed signal is to send it, in an unsigned format, to be stored in the block ram (BRAM).

Signed 18-bit




Unsigned



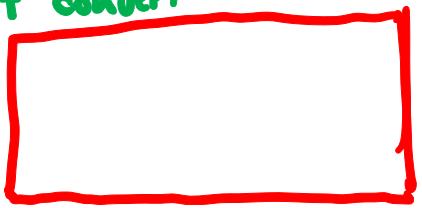

| Input Value | | Ouput Value | |
|---|---|---|---|
| 2's complement | 2's value | unsigned | unsigned value |
| 100...000 | -131072 | 000...000 | 0 |
| 111...111 | -1 | 011...111 | 131071 |
| 000...000 | 0 | 100...000 | 131072 |
| 000...001 | 1 | 100...001 | 131073 |
| 011...111 | 131071 | 111...111 | 262143 |

signed

TRICK?

sig_unsigned <= sig_signed with _____ or?

what does plot look like if you don't convert?



→ sim_plot_test_convert.jpg

# Lab 2 – Datapath

```
entity lab2_datapath is
    Port(       clk : in  STD_LOGIC;
                reset_n : in  STD_LOGIC;
                ac_mclk : out STD_LOGIC;
                ac_adc_sdata : in STD_LOGIC;
                ac_dac_sdata : out STD_LOGIC;
                ac_bclk : out STD_LOGIC;
                ac_lrclk : out STD_LOGIC;
                scl : inout STD_LOGIC;
                sda : inout STD_LOGIC;
                tmds : out  STD_LOGIC_VECTOR (3 downto 0);
                tmdsb : out  STD_LOGIC_VECTOR (3 downto 0);
                sw: out std_logic_vector(2 downto 0);
                cw: in std_logic_vector (2 downto 0);
                btn: in         STD_LOGIC_VECTOR(4 downto 0);
                exWrAddr: in std_logic_vector(9 downto 0);
                exWen, exSel: in std_logic;
                Lbus_out, Rbus_out: out std_logic_vector(15 downto 0);
                exLbus, exRbus: in std_logic_vector(15 downto 0);
                flagQ: out std_logic_vector(7 downto 0);
                flagClear: in std_logic_vector(7 downto 0));
end lab2_datapath;
```

switches?

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Flag Register

| reset_n | clk | set | clear | Q+ |
|---|---|---|---|---|
| 0 | X | X | X | 0 |
| 1 | 0,1,falling | X | X | Q |
| 1 | rising | 0 | 0 | Q |
| 1 | rising | 1 | 0 | 1 |
| 1 | rising | 0 | 1 | 0 |
| 1 | rising | 1 | 1 | X |

```
entity flagRegister is
        Generic (N: integer := 8);
        Port(     clk: in  STD_LOGIC;
                  reset_n : in  STD_LOGIC;
                  set, clear: in std_logic_vector(N-1 downto 0);
                  Q: out std_logic_vector(N-1 downto 0));
end flagRegister;
```

or N=1 is okay

UNITED STATES AIR FORCE ACADEMY

# VHDL Package file

- **Packages – Package for Lab 2**
- **Include this at the top of your file:**

  **use work.lab2Parts.all; -- all my components are declared here**

don't declare twice!

Do not need ac97.vhdl

• convert_unsigned_tb.vhdl

# VHDL Code

• sandbox_tb.vhdl

• sim_plot_test_convert.jpg

- Overall Lab 2 File: lab2.vhd
  - Lab 2 Datapath: Lab2_datapath_tb.vhd
  - Audio Codec Wrapper: Audio_Codec_Wrapper.vhd (Audio Codec Wraper for Xilinx Vivado)
    - i2s_ctl.vhd (I2S Transmitter portion of Audio Codec Wraper for Xilinx Vivado)
    - audio_init.v (Audio Initializer portion of Audio Codec Wraper for Xilinx Vivado) ← verilog?
    - TWICtl.vhd (TWI Controller portion of Audio Codec Wraper for Xilinx Vivado)
    - You need to add a clocking wizard for the Audio Codec and set the output frequencies to what is required (see comments in the Audio Codec Wrapper file).
- Constraint file: Lab2.xdc

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Generating Audio Waveforms

- Since you need to use a 3.5mm jack to input signals to the Nexys Video board, your phone's audio output works quite well. However, make sure you get an app where you can control both the left and right audio channels individually (i.e. the green and yellow signals in the figure above). The Keuwl Dual Channel Function Generator (available on Google Play) works well for Android Phones, and is easy to use once you get the hang of it.

?

Or use laptop's AUX port

UNITED STATES AIR FORCE ACADEMY

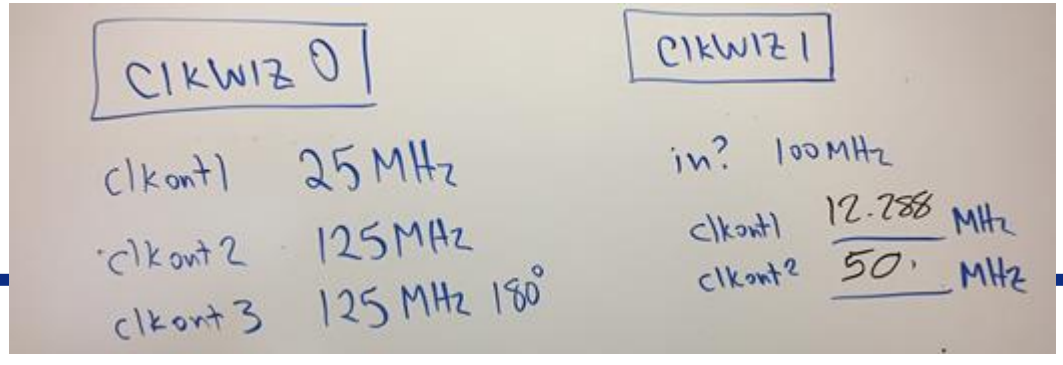
# Lab 2 – Requirements Gate Check 1

- **Gate Checks for Required Functionality**
  - There are 3 gate checks associated with this lab, each worth 5 points - see the rubric.
- **Gate Check 1**
  - By ~~**COB**~~ **Taps Lesson 13**, you must have started a Lab 2 Vivado project and downloaded the template files and drop in your Video, VGA, Scopeface, dvid, and tdms files from Lab 1 into your Lab 2 project in order to test your Lab 1 Scopeface works when you implement your BRAM using the two initialized signal examples in **~~BRAM_example_init~~.vhd**.
  - This does not require the audio or your control unit is working yet. Your **Scopeface/Video should continuously be reading the left and right BRAM signals displaying them on the monitor.**

Lab2_cu.vhdl
Lab2_dp.vhdl
already in datapath

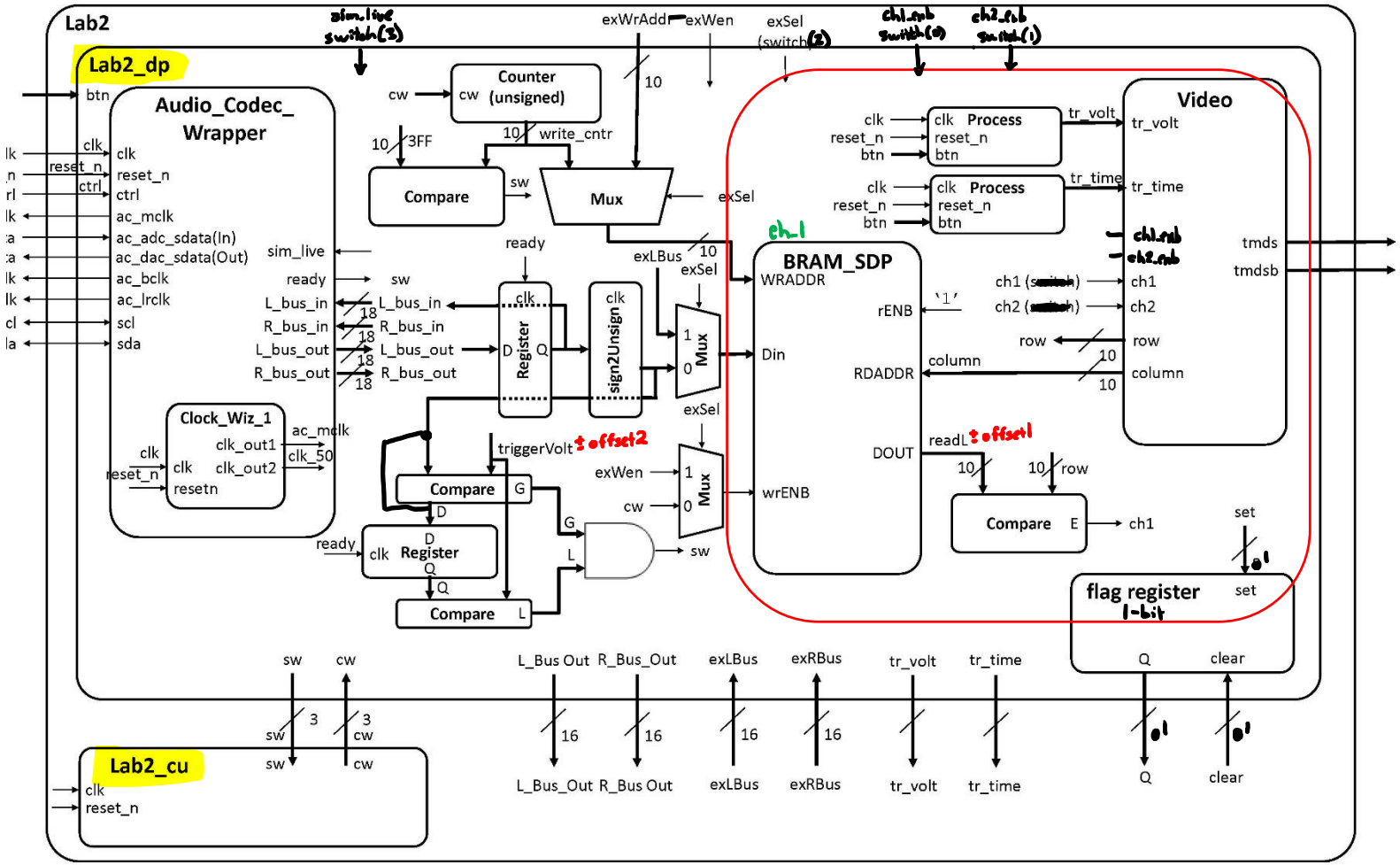
# Lab 2 – Requirements Gate Check 1

- **Gate Check 1**
  - You must implement Video entity (from Lab 1) to take the channel output from the left and right BRAMs and send it to the Channel 1 and 2 inputs to be displayed **when the readL and readR values equal the row value**. ±offset
  - Since there is no trigger, the waveform will be scrolling across the display and the scaling may be wrong
  - You will also have to re-implement the Lab 1 Clocking Wizard in you Lab 2 project.
  - Additionally your Scopeface and Button inputs from Lab 1 should be functional as well. and switches from lab1



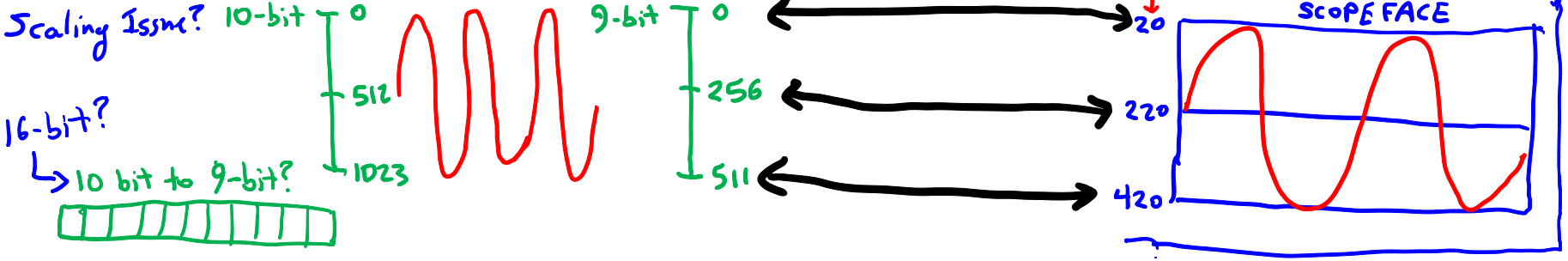
UNITED STATES AIR FORCE ACADEMY

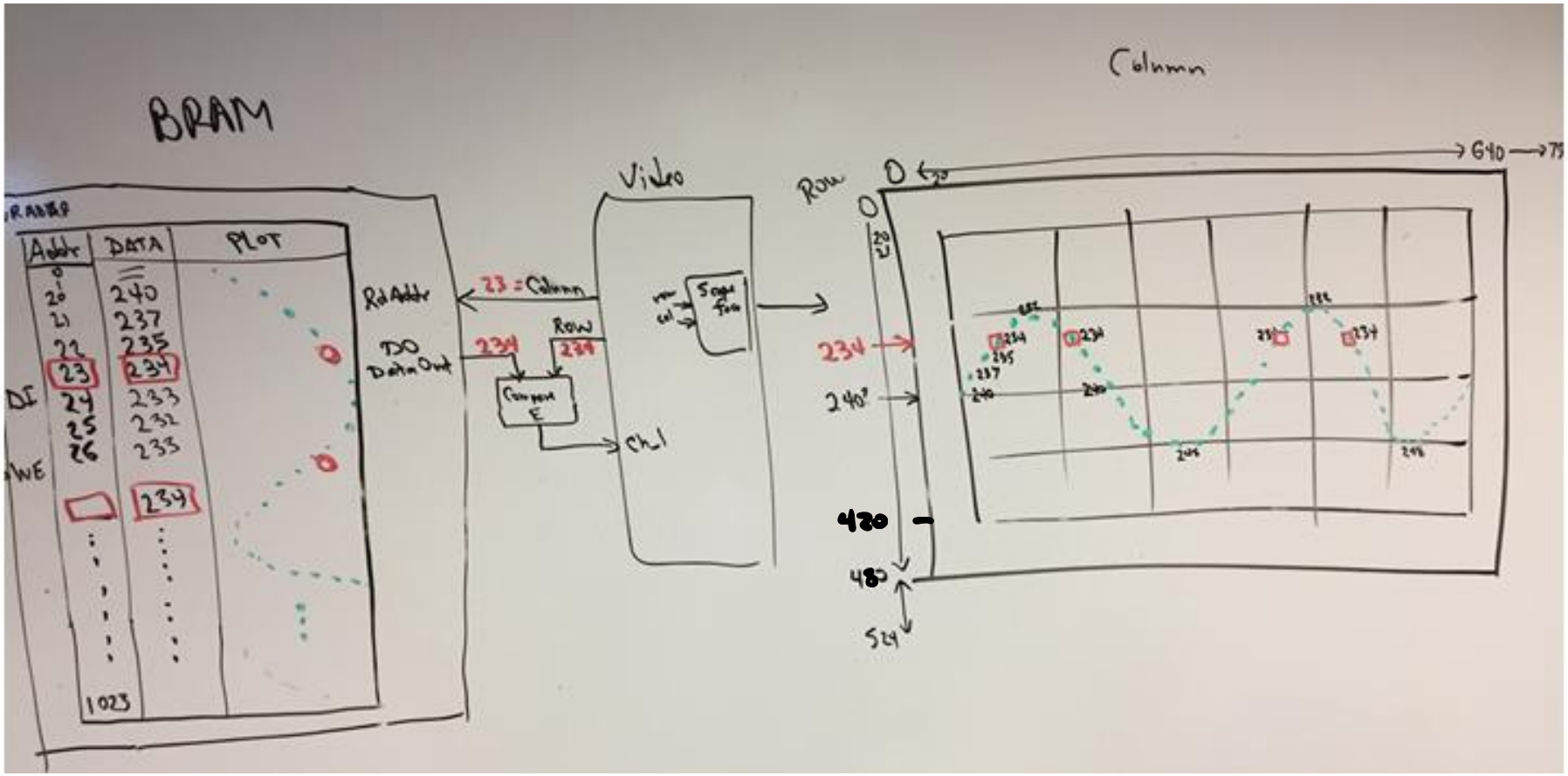
# Gate Check 1




SAMPLES FROM BRAM

ROW


Scaling Issue? 10-bit
16-bit?
10 bit to 9-bit?
0
512
1023
9-bit
0
256
511
20
220
420
SCOPE FACE

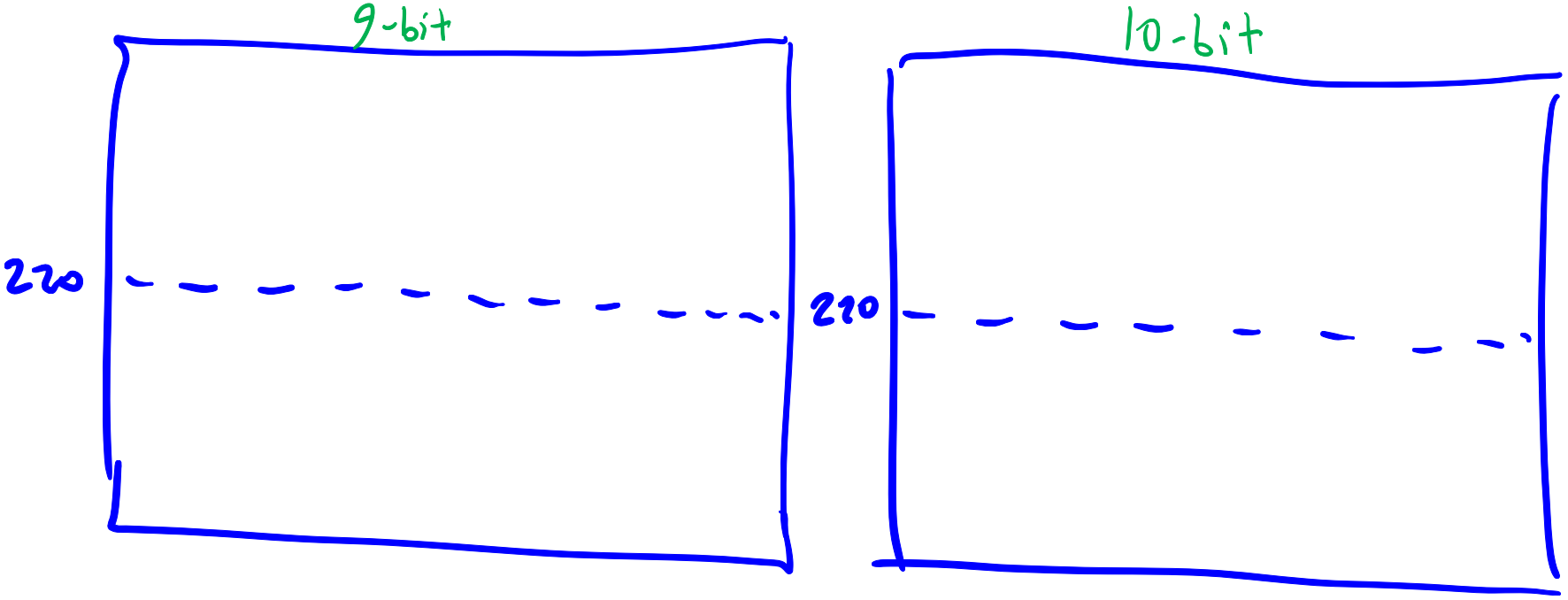

BRAM
Addr
DATA
PLOT
240
237
235
234
233
232
233
234
1023
RdAddr
DO
DataOut
23 = Column
234
Row
234
Compare
Ch1
Video
Row
Column
0
234
240
420
480
524
640

9-bit Offset = 256-220 = ____

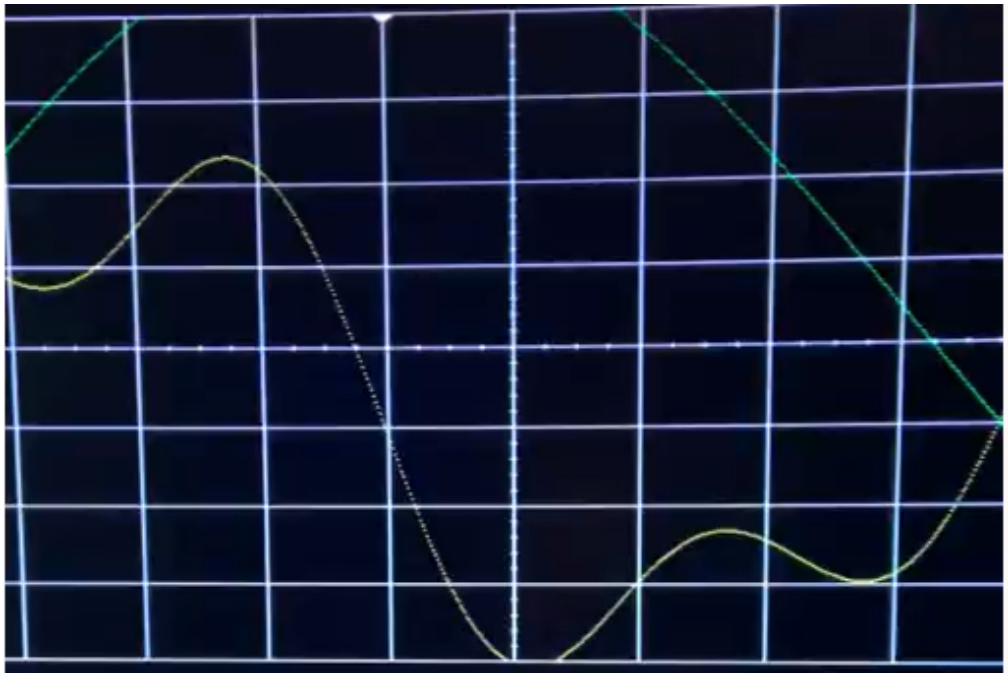
10-bit Offset = 512-220 = ____

chl <= '1' when (row = readL ± offsetl) else '0';

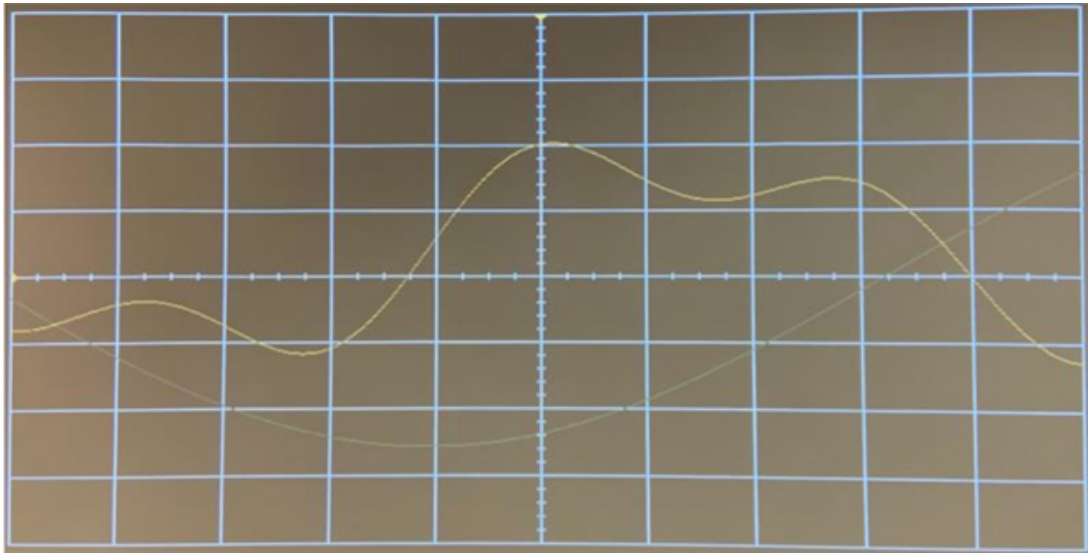
what if don't adjust for offset?



# BRAM TEST DATA on SCOPEFACE



← 10-bit Data



← 8-bit Data In a 10-bit word
(Because "row" is 10-bits)

Because we don't see the full cycle of the sinewave, it is hard to know if they got the DC bias exact

ECE 383 2022 Here is a hack to test if you removed the DC bias (offset) correctly:
Our 16-bit values range in hex from 0000 to FFFF, so 8000 is the middle which should correspond to row 220.
x8000 = 1000 0000 0000 0000 (16-bit, unsigned)
= 1000000000 (10-bit = 512)
or 100000000 ( 9-bit = 256)
or 10000000 ( 8-bit = 128)

So, to test the bias offset, for one of the channels, disconnect the BRAM, and instead change your CSA comparitor to something like:

```
ch1 <= '1' when ("1000000000" +/- offset = row) else '0';  -- if 10-bit
or
ch1 <= '1' when ("100000000" +/- offset = row) else '0'; -- if 9-bit
or
ch1 <= '1' when ("10000000" +/- offset = row) else '0'; -- if 8-bit
```

* The syntax might need some work [stdlogicvectors vs unsigned]
This should draw a yellow or green line along row 220 if you get the offset correct

draws a straight line

or see Sandbox_tb.vhdl

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Gate Check 2

- **Gate Check 2**
  - NOTE: THIS IS THE HARDEST PART! By **COB** **Lesson 15**, you must have implemented and connected the BRAM Address Counter to left and right BRAMs, instantiated the Audio Codec Wrapper in Simulation mode (**sim_live = '0'**), and your control unit, such that your control unit writes the simulated audio data to the left and right BRAM and you can see the waveforms plotted on the monitor.
  - Since there is no trigger, the waveform will be scrolling across the display
  - convert the signed audio data from the Audio_Codec_Wrapper into unsigned data
  - implement another Clocking Wizard for the Audio Codec

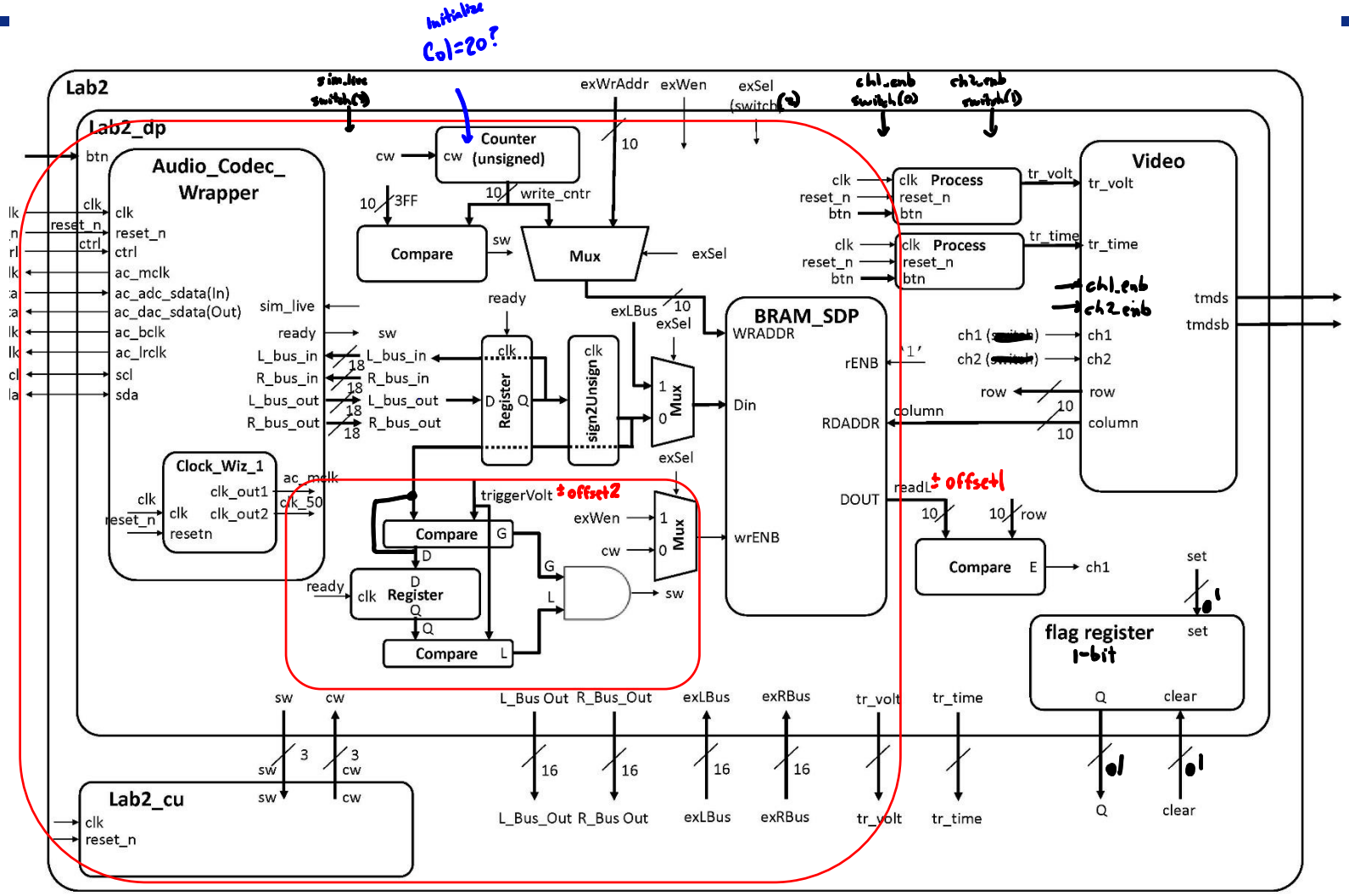
taps

← unless your BRAM write stays in sync with Codec each cycle (like both done 1024 samples)

if you want to freeze the scrolling without a trigger, have your FSM only write one frame of 1024 samples to BRAM, then stop. Scopeface will then continually plot this same data over-and-over.

UNITED STATES AIR FORCE ACADEMY

# Gate Check 2
# Simulated Audio

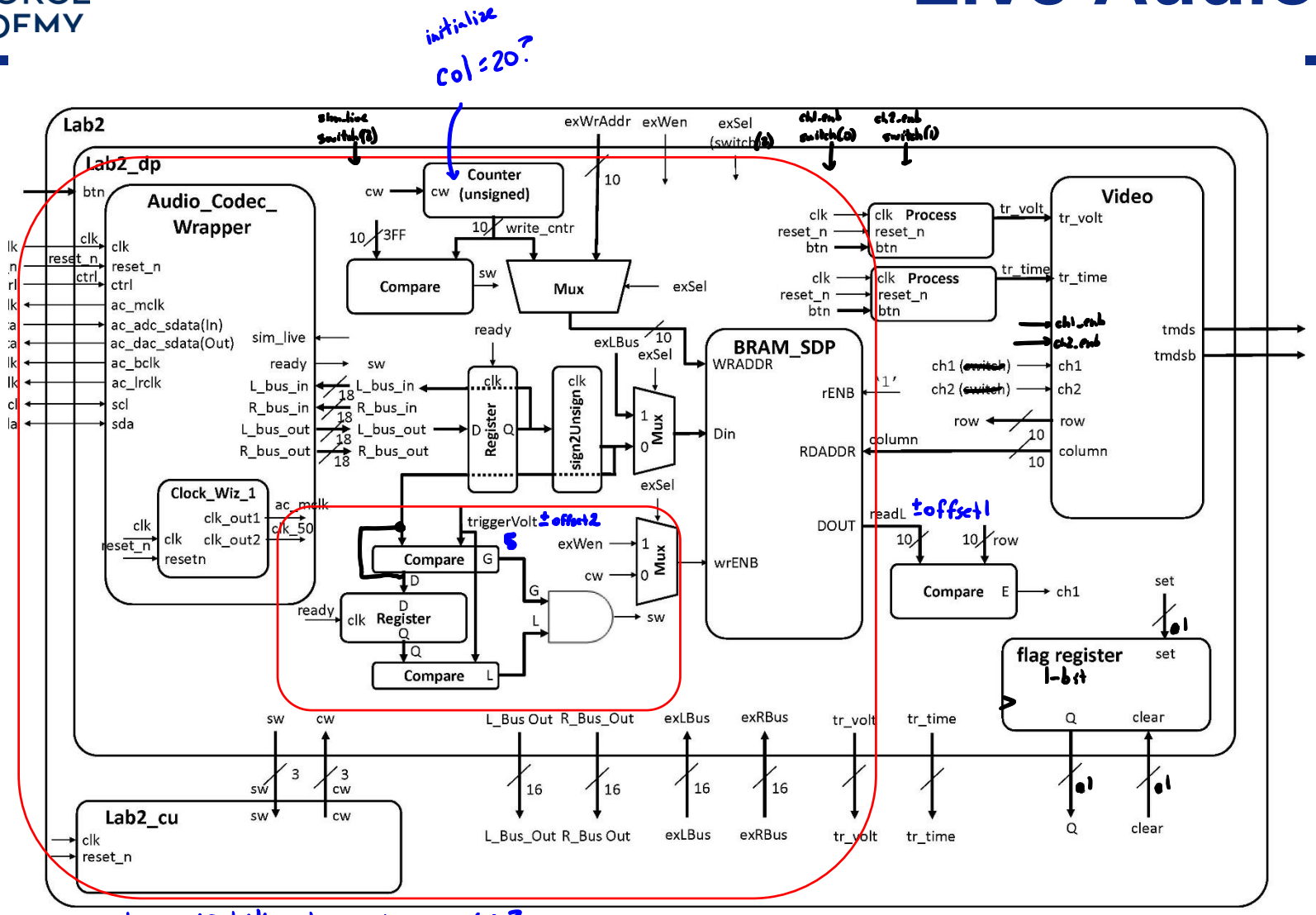


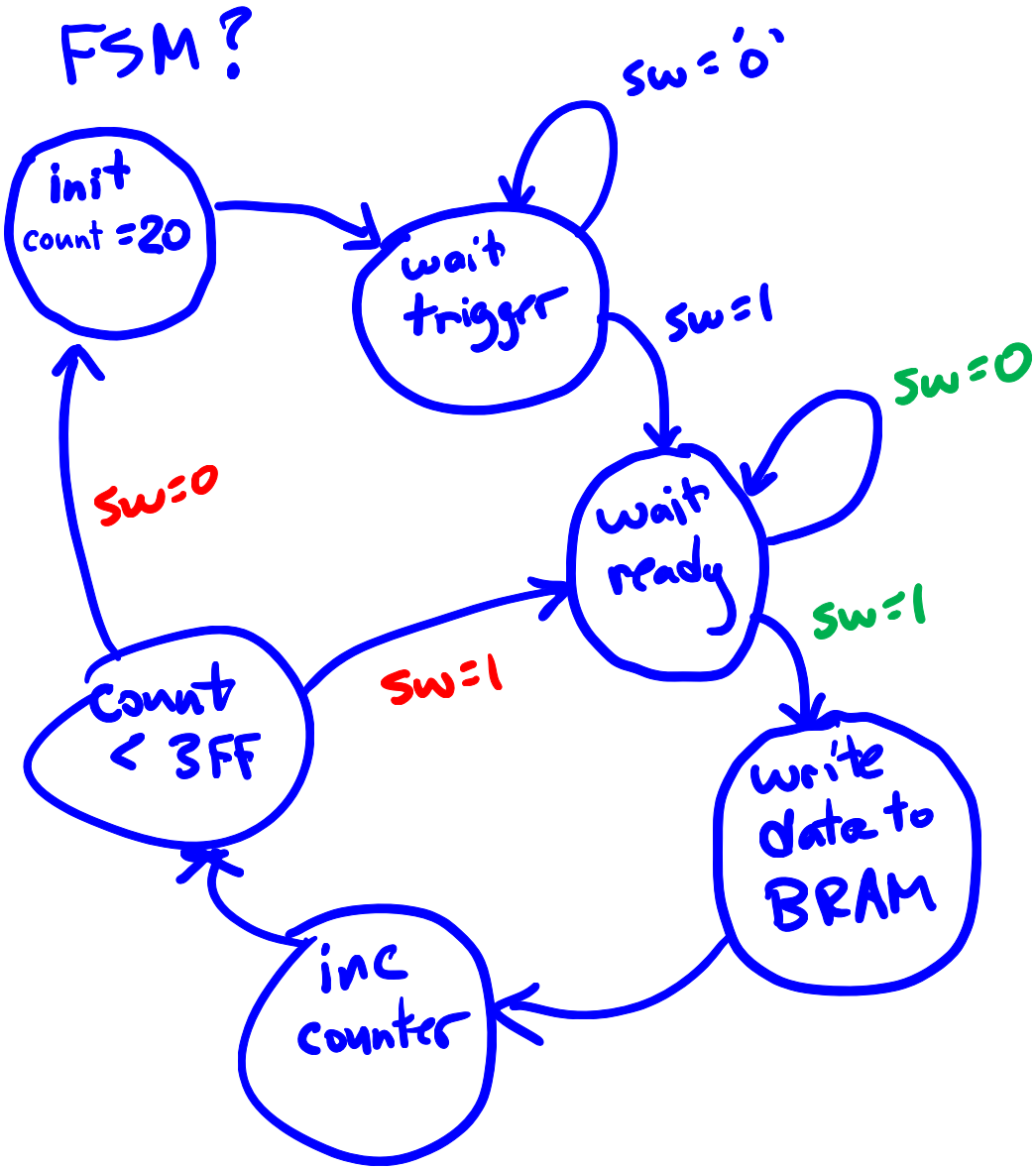
# Lab 2 – Requirements Gate Check 3

- **Gate Check 3**
  - By ~~COB~~ **Lesson 16**, redo Gate Check 2, except with the Audio Codec Wrapper in Live mode (**sim_live = '1'**).
  - since there is no trigger, the waveform will be scrolling across the display
  - Also make connections to loopback the serial ADC input back out to the DAC output (i.e. send the signal back into the Codec).
  - After you finish Gate Check 3, this is a good time to implement proper triggering on the trig_volt value.
  - If this does not work, you must create a Testbench to help debug why it is not working.

UNITED STATES AIR FORCE ACADEMY

# Gate Check 3
# Live Audio






what if you
don't initialize the counter to 20?

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Required Functionality

- **Required Functionality**
  - Get a single channel of the oscilloscope to display with reliable triggering that holds the waveform at a single point on the left edge of the display. A 220Hz waveform should display something similar to what is shown in the screenshot at the top of this page. Additionally, you must have the following done:
    - ~~Use a package file to contain all your component declarations.~~
    - Use separate datapath and control unit.

trigger not calibrated

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements
# Required Functionality Cont 1

- **Required Functionality**
  - Your datapath must use processes (or CSA) which are similar to our basic building block (counter, register, mux, etc.). I do not want to see one massive process that attempts to do all the work in the datapath.
  - Mini-C not required
  - ~~Testbench for the flagRegister.~~
  - ~~Testbench for the control unit.~~

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Required Functionality Cont 2

- **Required Functionality**
  - ~~Testbench for the datapath unit showing data (different value than what is given in the testbench) coming out of the audio codec and being converted from signed to unsigned and then to std_logic_vector to go into your BRAM. Include calculations to back up what the waveform shows.~~

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Required Functionality Cont 3

- **Required Functionality**
  - ~~For Bonus Points: Testbench for the datapath unit showing that same data coming out of the BRAM. Make sure you show the read address and the data values coming out. This will require you to set your control words on the testbench. Additionally, you will have to drive the pixel_clock on the Video Module. Once you get the datapath testbench running you will notice that DCM module doesn't put out a clock in the Video Module.~~

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements B-Level Functionality

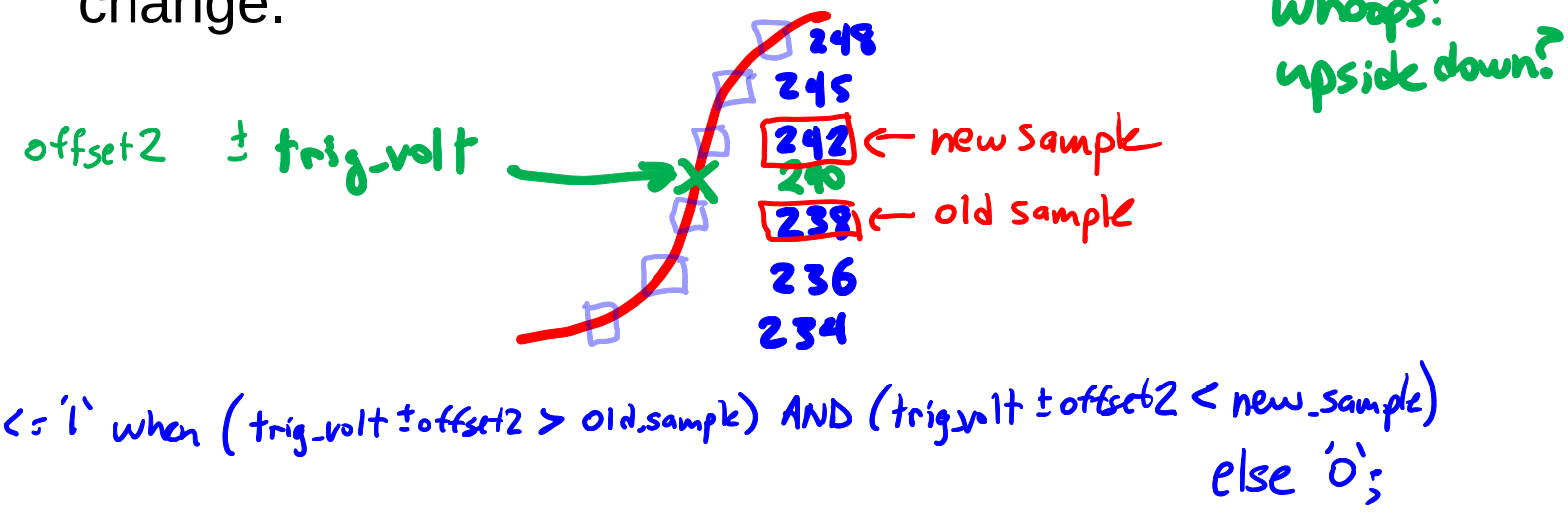
- **B-level Functionality** (You can swap A and B functionality)
  - Meet all the requirements of required functionality
  - Add a second channel (in green).
  - Integrate the button debouncing strategy in HW #7 (or another equivalent debouncing method) to debounce the buttons controlling the trigger time and trigger voltage.
  - Move the cursors on the screen.

Moved to A functions

- Include flag register and exSel Muxes

# Lab 2 – Requirements A-Level Functionality

- **A-level Functionality** • deBounce
  - Meet all the requirements of B-level functionality.
  - Use the trigger voltage marker to establish the actual trigger voltage used to capture the waveform. As the trigger is moved up and down, you should see the point at which the waveform intersects the left side of the screen change.



UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Turn In

- **Turn In Requirements**
  - All your work in this lab is to be submitted using Bitbucket. The main part of the lab is your README, documenting your design. Your README must include the following:
  - **Introduction** - Provide a brief overview of the problem.
  - **Implementation** - Provide block-diagram of your solution using the **signal names in your code**. The block diagram given above is somewhat incomplete, so make sure to include corrections to it. For each module that you built, explain its overall purpose, inputs, outputs, and behavior. Include all your vhdl files (code and testbench), wcfg file, and bit files. Put these in a folder called "code".

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Turn In Cont 1

- **Turn In Requirements**
  - **Test/Debug** - Briefly describe the methods used to verify system functionality.
  - List the major problems you encountered and how you fixed them. This should cover all the problems you encountered in the lab and how you fixed them. Break each problem and solution into separate paragraphs.
  - ~~**Capability** - Well you have built a oscilloscope, what are its capabilities?~~
    - ~~The horizontal axis represents time. There are 10 major divisions on the display; how long does each major division represent?~~

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Turn In Cont 2

- **Turn In Requirements**
  - **Capability Continued -**
    - ~~Each major time division is split into 4 minor division, how long does each minor division represent?~~
    - ~~Generate a sine wave that can be fully captured on your display (like the yellow channel in the image at the top of this web page). record its height in major and minor vertical divisions. Measure this same audio output using the break out audio cable. Record the peak-to-peak voltage. Compute the number of volts in each major and minor vertical division.~~

UNITED STATES AIR FORCE ACADEMY

# Lab 2 – Requirements Turn In Cont 3

- **Turn In Requirements**
  - **Capability Continued** -
    - ~~Starting at address 0, how long does it take to fill the entire memory with audio samples (coming in at 48kHz)?~~
    - ~~How long does it take to completely draw the display once?~~
    - ~~The question is likely relevant to Lab 3 - how long is the vsynch signal held low?~~
  - **Conclusion** - Explain what your learned from this lab and what changes you would recommend in future years to this lab or the lectures leading up to this lab.