EES4725: DIGITAL CIRCUITS AND FPGA DESIGN

FPGA Design Project:
Sound Display and Entertainment System

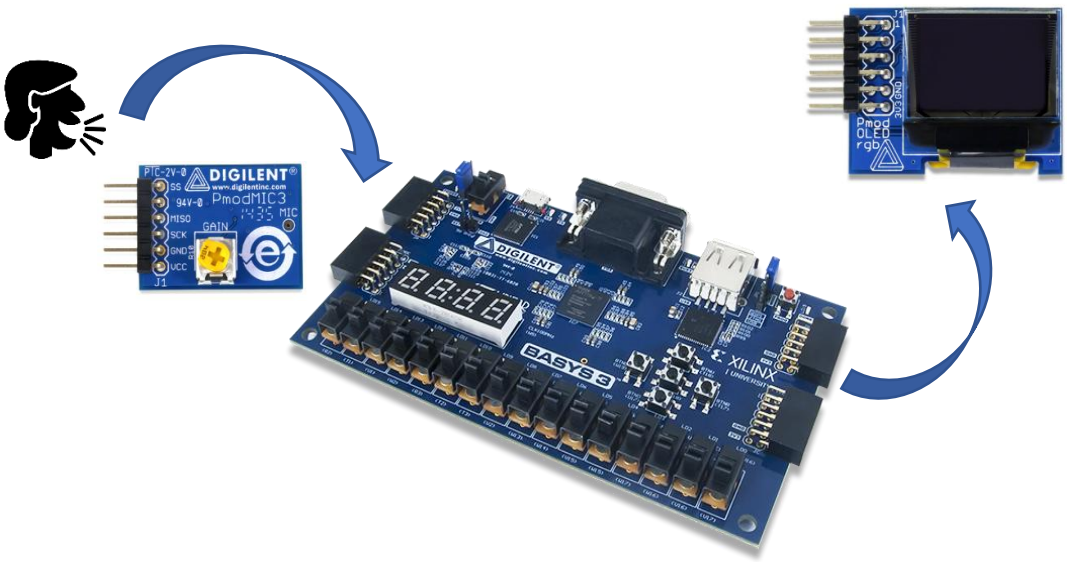
ABSTRACT

Using the fundamental technical skills obtained from the previous lab sessions, you will be creating a Sound Display and Entertainment (SDE) system. Two additional devices will be provided to you for the system:

- **MEMs microphone:** This analog-to-digital device will capture audio signals from the surroundings, and provides the digitized audio data to you in a 12-bit format.
- **OLED RGB Display Screen:** Information can be displayed on this 96 x 64 bit display screen with 16-bit colour resolution.

(You will need to replace the devices at your own cost if damaged. Use them with upmost care!)

This manual introduces to you some concepts involved, and guides (NOT handhold!) you through getting a basic SDE up and running. You will need to use your own logic, creativity and resourcefulness to enhance the SDE system.



1. PROJECT OVERVIEW

This SDE system is a pair work project. Some milestones that you can start with for the SDE system are detailed in **Section 3** of this manual. To make your system more functional, user-friendly and unique, your team will also need to work on features of your own choosing!

2. HARDWARE AND SOFTWARE RESOURCES

HARDWARE COMPONENTS:

- Each team will sign out **1 PmodMIC3** and **1 Pmod OLEDrgb**
- Each student has already been assigned **1 Basys 3** Development Board

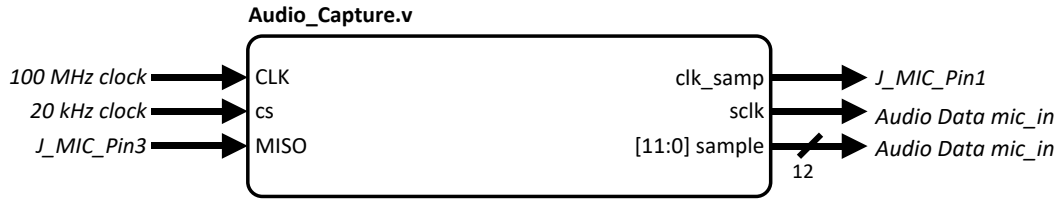
SOFTWARE FILES:

A project template (**SoundDisplay.xpr.zip**) can be downloaded. The template consists of the following:

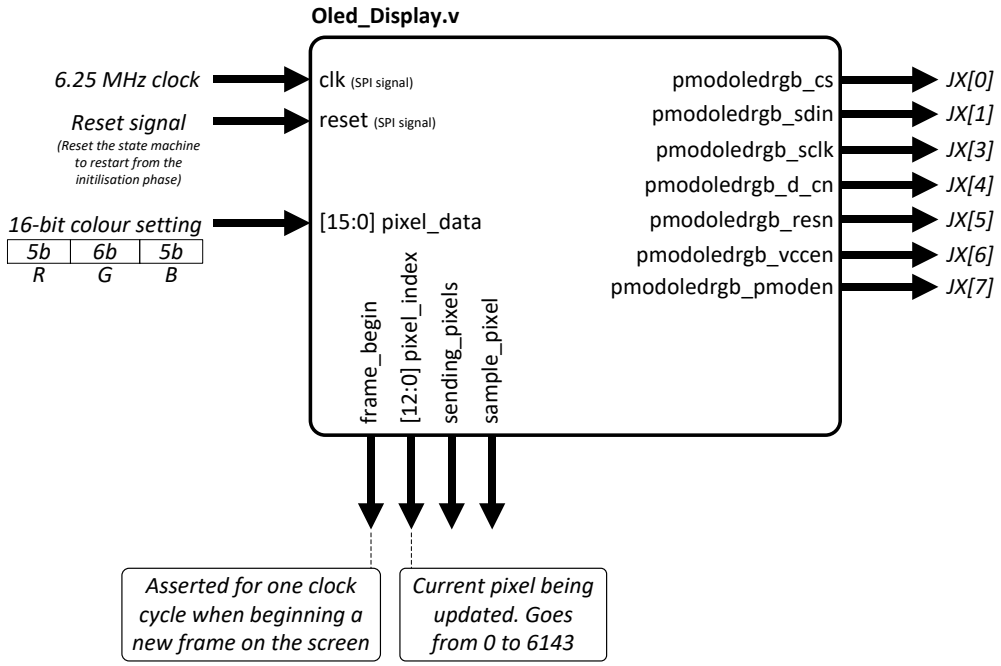
Design Sources:

Top_Student.v: The top-module of the design. This will be your main module, or typically called the Top Level module, where you instantiate the sub-modules and make the necessary links between these sub modules. **You will need to modify this module, as well as create other design sources for use in this module.**

Audio_Capture.v: An interface module between the microphone pmod device and your design on the FPGA. This module works with the *PmodMIC3* to convert the serial data input into a 12-bit parallel `mic_in` sample data. The conversion needs a sampling clock and a serial clock. **You are NOT required to make changes to this module.**



Oled_Display.v: An interface module between your design on the FPGA and the OLED display. This module works to send serial SPI data into the OLED display for initialization and drawing. **You are NOT required to make changes to this module.**

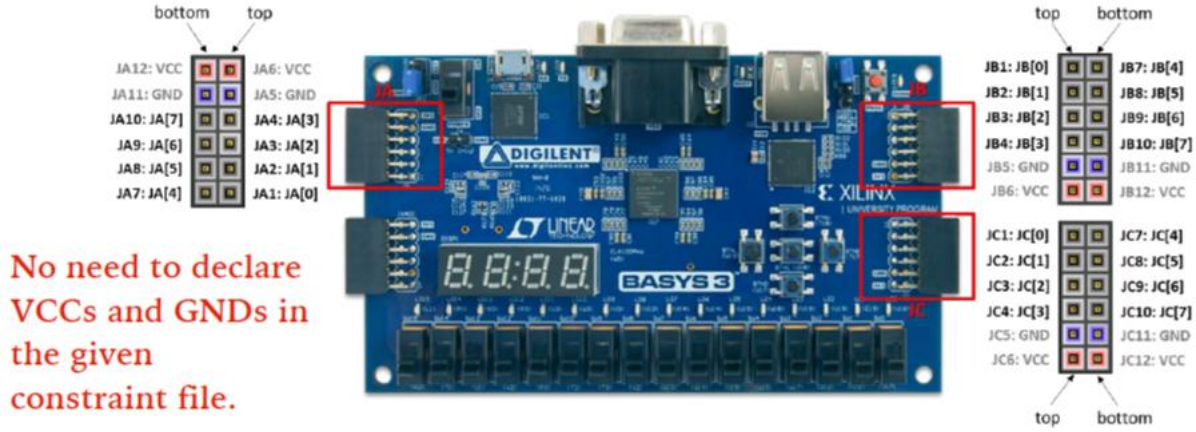


3. TASKS AND REQUIREMENTS

3.1 BASYS3 PMOD PORTS

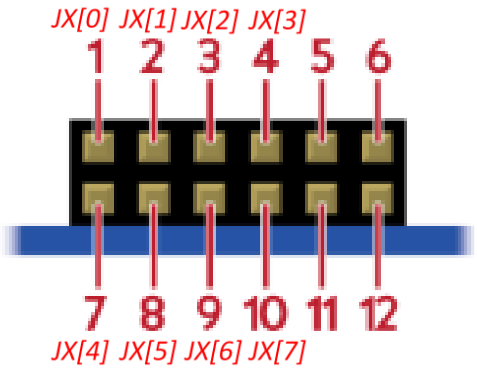
OBJECTIVE: To understand the pmod ports on the Basys 3 development board.

The Basys 3 has 4 double row Pmod ports that can be used as simple expansion ports. Digilent produces a large collection of Pmod (Peripheral Module) accessory boards that can attach to the expansion ports to add ready-made functions such as A/D's, D/A's, motor drivers, sensors, displays, and many other functions.



The port mappings are provided in the constraints file as shown below. Please take note that VCCs and GNDs do not need to be declared in the constraint file.

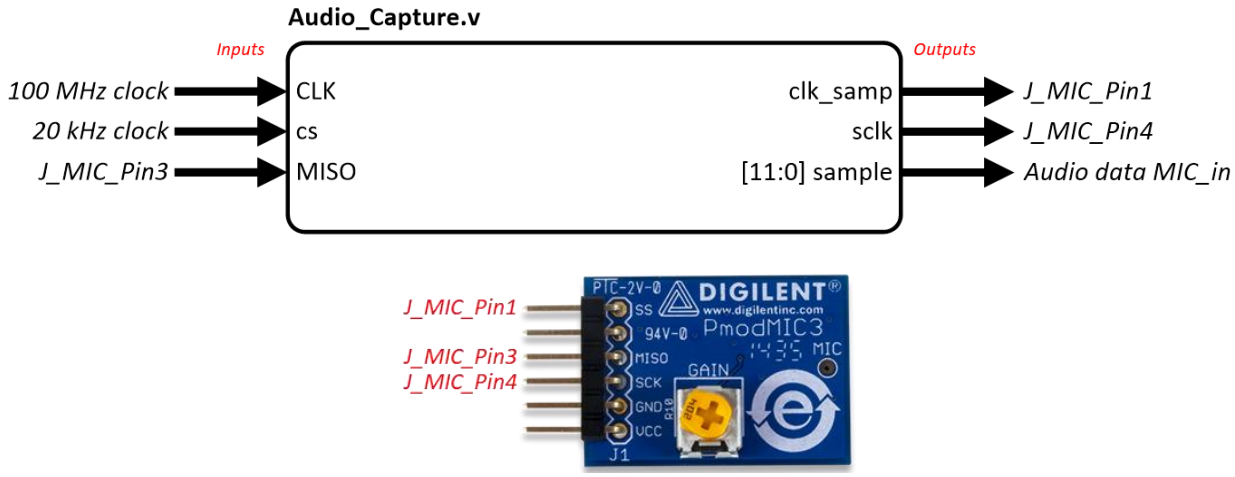
```
##Pmod Header JA
##Sch name = JA1
#set_property PACKAGE_PIN J1 [get_ports {JA[0]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[0]}]
##Sch name = JA2
#set_property PACKAGE_PIN L2 [get_ports {JA[1]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[1]}]
##Sch name = JA3
#set_property PACKAGE_PIN J2 [get_ports {JA[2]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[2]}]
##Sch name = JA4
#set_property PACKAGE_PIN G2 [get_ports {JA[3]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[3]}]
##Sch name = JA7
#set_property PACKAGE_PIN H1 [get_ports {JA[4]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[4]}]
##Sch name = JA8
#set_property PACKAGE_PIN K2 [get_ports {JA[5]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[5]}]
##Sch name = JA9
#set_property PACKAGE_PIN H2 [get_ports {JA[6]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[6]}]
##Sch name = JA10
#set_property PACKAGE_PIN G3 [get_ports {JA[7]}]
    #set_property IOSTANDARD LVCMOS33 [get_ports {JA[7]}]
```



3.2 SYSTEM SET UP - BASIC DISPLAY OF AUDIO SIGNAL ON OLED

OBJECTIVE: To set up the SDE system, by interfacing the Basys 3 Development Board with the PmodMIC3 and the PmodOLED. Your system will capture and digitize the audio signal input from the microphone on the PmodMIC3. The data will then be processed by the FPGA, for display on the PmodOLED

STUDENT A: Setting up the Microphone



A1. **Instantiate the** `Audio_Capture.v` **module in** `Top_Student.v`

A2. **Create and provide the 100MHz and 20kHz (name it** `clk20k` **) clock signals to** `Audio_Capture.v`
The 20kHz clock is used as the sampling clock for capturing the audio signal. At every rising edge of this 20kHz clock signal, the audio signal from the microphone is read and stored in a 12-bit register named `sample`. This means that a 1s audio signal will need to be represented by 20,000 discrete samples.

A3. **Name the microphone data received from** `Audio_Capture.v`, `mic_in` **and display this on 12 LEDs.**
`mic_in` is the 12-bit data that is being read by the microphone. By displaying the microphone input on the LEDs, a quick visual observation can indicate if the microphone is properly connected.

A4. **Connect the rest of the signals (**`J_MIC_Pin1/3/4`**) to the PmodMIC3 device accordingly.**

A5. **Attach the PmodMIC3 to the TOP ROW of the pmod port.**
Update the constraints file according to the chosen Pmod port (eg. JA / JB / JC).
Very Important: To avoid damaging the boards, make sure the GND and VCC pins on the Pmod and Basys3 are connected correspondingly

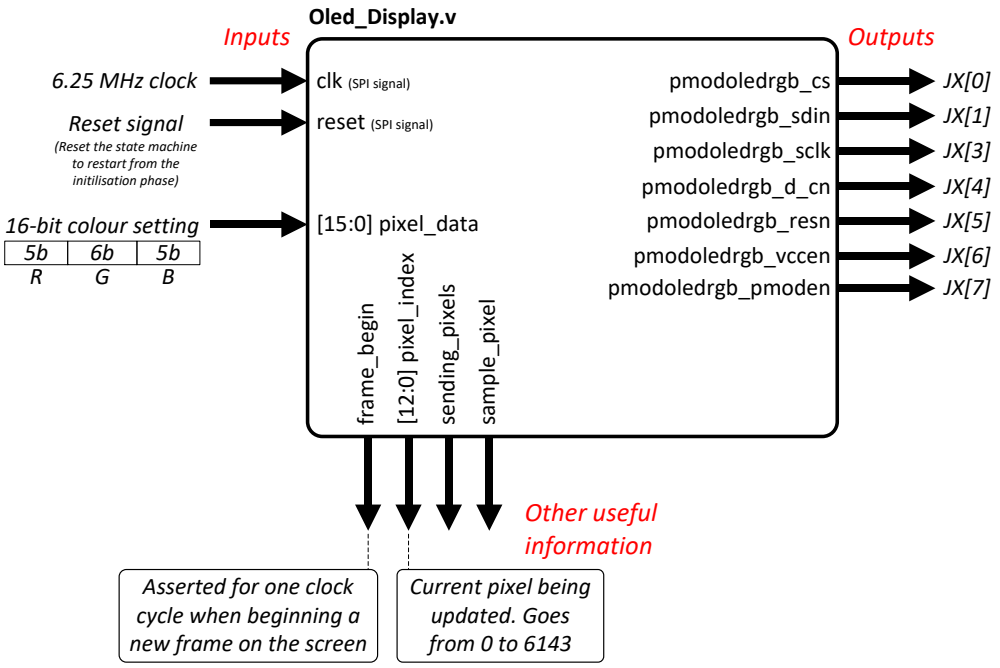
A6. If the above steps are executed correctly, you would observe 12 LEDs flicker at a very high speed.

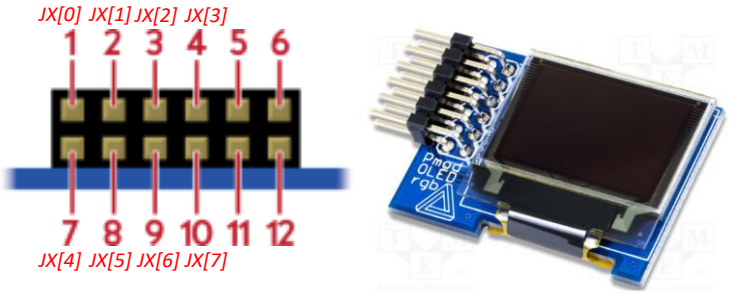
A7. For easier observation, implement a switch that allows you to switch between one of these two options:

- sw[0] is OFF: The 12-bit `mic_in` is represented on LEDs that updates at a clock speed of 20 kHz
- sw[0] is ON: The 12-bit `mic_in` is represented on LEDs that updates at a clock speed of 10 Hz

If the above steps are executed correctly, you would observe 12 LEDs flicker at a very high speed when sw[0]is OFF, and at a more observable speed when sw[0] is ON.

STUDENT B: Setting up the OLED





B1. **Instantiate the** `Oled_Display.v` **module in** `Top_Student.v`

B2. **Create and provide a 6.25MHz clock signal named** `clk6p25m`**, to** `Oled_Display.v`

B3. **Create a 16-bit signal named** `oled_data` **and initialize it with a value of** `16'h07E0`**.**
The RGB OLED screen has 16-bit color resolution. This 16-bit RGB resolution is represented through 5 bits for the red colour component, 6 bits for the green colour component, and 5 bits for the blue color component.
Connect `oled_data` to the `pixel_data` input of `Oled_Display.v`

B5. **Connect the** `JX[0:7]` **signals to the PmodOLED device accordingly.**

B6. **Attach the OLED display to the Basys 3 and update the constraints file accordingly.**

B7. **Give the** `reset` **port a constant value of 0.**

B8. **Generate the bitstream and download to the FPGA**
Verify that the background colour of the screen is green. Why is it green?

TEAMWORK: System Integration

C1. **Connect the MSBs of `mic_in` to part of the 16-bit colour setting given to the OLED through `oled_data`**
The 16-bit colour setting consists of 5 bits for the red colour component, 6 bits for the green colour component, and 5 bits for the blue component. According to the **first rightmost numerical digit** of the matriculation card of **student B**, the 5 (or 6 for green) most significant bits of `mic_in` must be connected to the RGB components as tabulated below:

Rightmost numerical digital of student B	R	G	B
0	✔		
1		✔	
2			✔
3	✔		✔
4	✔		
5		✔	
6			✔
7	✔		✔
8		✔	
9			✔

A tick (✔) indicates that the 5 (or 6 for green) most significant bits of `mic_in` are connected to that respective colour component

C2. **Generate the bitstream and download to the FPGA.**

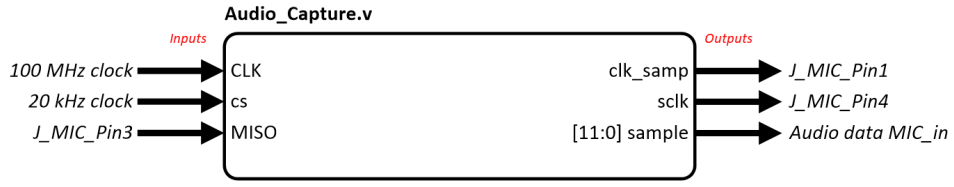
C3. **If you are able to observe the following, it indicates that the microphone and OLED screen has been set up successfully.**

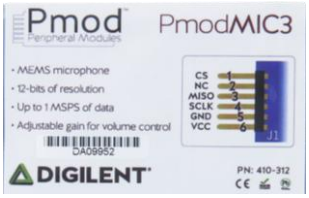
- Provide an audio signal to the microphone, you should observe a single colour with varying intensities, depending on the audio signal
- Using a tone generator application, supply a sine wave to the microphone. You will observe a cyclic repetitive pattern on the OLED screen.

3.3 SYSTEM BASIC FEATURES -

[STUDENT A] EXTRA NOTES and SUGGESTED FEATURE

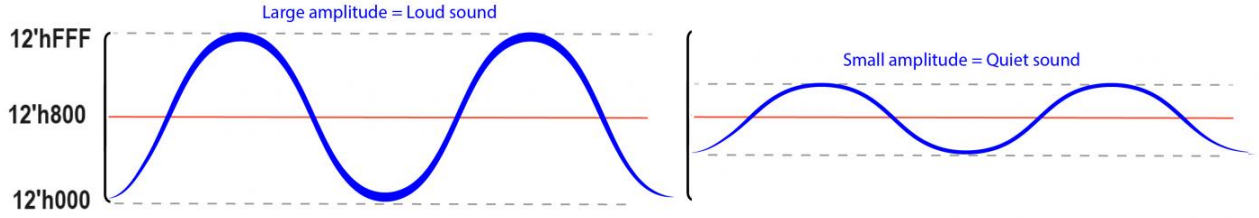
Based on the above task, you would have successfully completed setting up the microphone on the PmodMIC3 board. This section provides more details on how it works. For more details, please refer here.





The MIC3 pmod board receives and digitizes the analog audio input signal received from the microphone, producing a 12-bit serial data through the MISO port. The Audio_Capture.v module parallelizes the serial data and produces a 12-bit parallel `mic_in` output that is updated at 20kHz. At this sampling frequency, audio signals of up to 10kHz Nyquist frequency can be digitized. However, it is recommended to operate below the Nyquist frequency.

For your reference, the human voice frequency range is approximately below 300Hz and the human hearing range goes from about 20 Hz to 20 kHz.

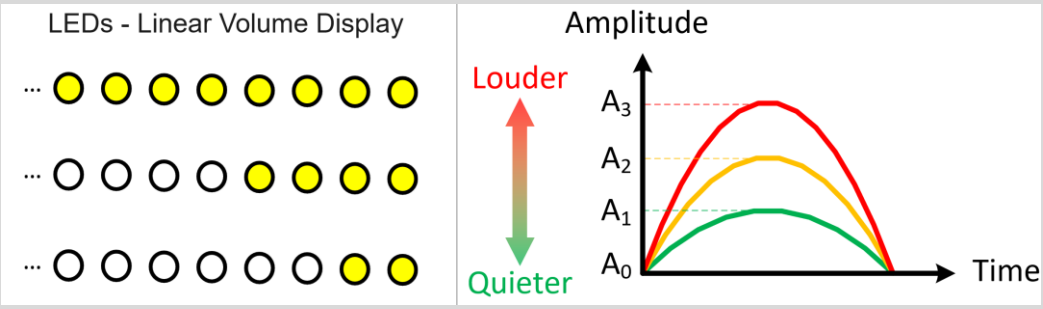


Please also take note that the `mic_in` data is biased at approximately the mid-point of the 12-bit signal range (~12'h800) as shown above.

SUGGESTED BASIC FEATURE :

VOLUME INDICATOR

The volume of an audio signal helps people to observe the amplitude of an audio signal visually. An audio volume indicator can be implemented by extracting the amplitude data from an audio signal, and displaying this data on LEDs.

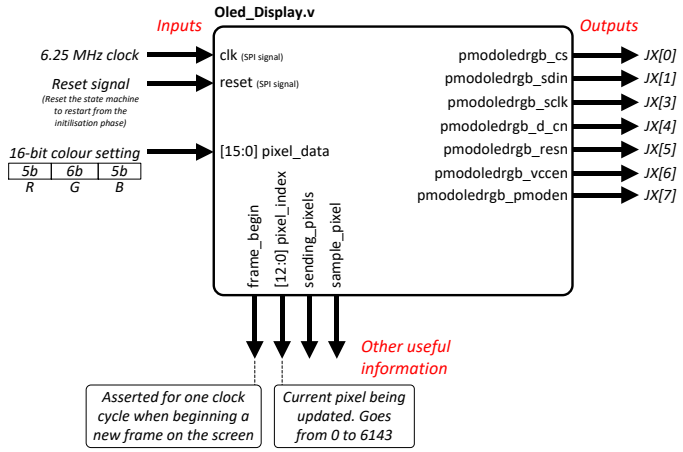


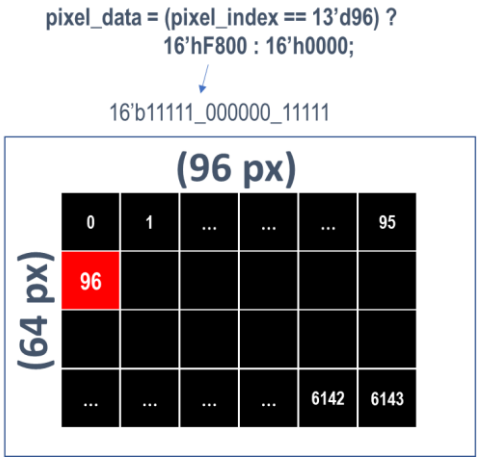
In this basic feature, you can use the 16 LEDs (or 7-segment display) on the Basys 3 as audio volume indicatorsEach LED, would represent an amplitude range from the audio signal.

To test if your volume extraction algorithm is working well, provide a sinusoidal test signal (link here) of constant amplitude as an input audio signal. For a sinusoidal test signal of constant volume, an approximately constant volume reading should be produced.

[STUDENT B] EXTRA NOTES and SUGGESTED FEATURE

Based on the above task, you would have successfully completed setting up the OLED screen on the Pmod OLEDrgb board. This section provides more details on how it works. For more details, please refer here.





The OLEDrgb is a display screen that has 96 pixels horizontally and 64 pixels vertically. The screen has a total of 6144 pixels and each pixel has a 16-bit colour resolution.

To control the display, the two important signals for Oled_Display.v are :
`pixel_index[12:0].` : provides the position of current pixel being updated and will output a value of 0 to 6143, from left to right and top to bottom.
`pixel_data[15:0]` : controls the colour of the current pixel being updated and has a 16-bit value representing the intensity of the Red (most significant 5 bits), Green (middle 6 bits), and Blue (least significant 5 bits) values.

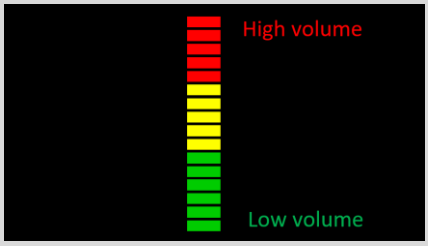
For example, the code above will display the image on the right where one pixel at position 96 will be displayed at the maximum redness intensity. For easier control of the OLED display, it is recommend that you work on creating an **x** and **y** coordinate system derived from `pixel_index` to make the pixel drawings less cumbersome.

- Hint : x = pixel_index % 95 (% is a modulus operator and outputs the remainder of the division)
- Hint : y = pixel_index / 63

SUGGESTED BASIC FEATURE :

VOLUME BAR DISPLAY

The volume of an audio signal helps people to observe the amplitude of an audio signal visually.



In this task, you need to design a volume bar with 16 levels and three different colours to display the volume information from Student A to improve the user-friendliness of the SDE system.

[STUDENT X and Y] CHALLENGING FEATURES - HANDWRITTEN DIGIT RECOGNITION

For students who prefer a challenge, you may explore working on a different project that involves the OLED screen, mouse and a convolutional neural network based, handwritten digit-recognition drawing app.

The project template (MODS.xpr.zip) provided consists of the following design sources :

Design Sources:

- Top_Student.v: The top-module of the design. This will be your main module, or typically called the Top Level module, where you instantiate the sub-modules and make the necessary links between these sub modules
- MouseCtl.vhd: An interface module between the computer mouse, Ps2Interface.vhd and your design on the FPGA. This module signals when clicks and movements are detected on the computer mouse connected to the Basys3 USB port. A wired mouse is recommended, as not all wireless mice will work. You are note required to make changes to this module. (Optionally, you should read the description inside if you want to know more about the input and output signals.)
- Oled_Display.v: An interface module between your design on the FPGA and the OLED display. This module works to send serial SPI data into the OLED display for initialization and drawing. You are not required to make changes to this module.
  (Optionally, you should read the description inside if you want to know more about the input and output signals.)
- Paint.v: A module that connects the computer mouse and OLED, so that digits created by the mouse on the OLED can be recognised, and the results shown on the seven segments display and LEDs on the Basys3. You are not required to make changes to this module. This module is optional, and its usage will not be explained.

The Paint App module is a canvas-based drawing app with a built-in handwritten-digit recognition using a simplified convolutional neural network based on the mnist handwritten digit image recognition dataset. If you'd like to explore building it, you can refer to

EES4725 >> Project >> Handwritten Digit Recognition >> "MODS.xpr.zip" and "EES4725 - Design Project - Handwriting Recognition Instructions.pdf" and "Demo Video"

4 SYSTEM IMPROVEMENTS

Working together with your partner, you can further implement improvement feature(s) to add value to the Sound Display and Entertainment system, in order to distinguish your unique system from the rest. The system can be made more interactive, interesting, functional and user-friendly.

The additional improvement(s) are open-ended and will be evaluated on the metrics of
(1) Quality
(2) Complexity
(3) Functionality
(4) Creativity

Before you connect other external devices to the Basys 3 development board, please obtain approval of the instructor.

5. PROJECT SUBMISSION

There are two items (ITEM A, and ITEM B) to be submitted.

5.1 ITEM A: PROJECT ARCHIVE SUBMISSION

- Only **ONE** Vivado project archive (.zip) per team. **The archive should not exceed 50 MB in size.**
- Ensure that your bitstream has been successfully generated and tested on your Basys 3 development board **BEFORE** submission.
- Name your file as Project_<Student_1_Name>_<Student_2_Name>.zip Eg. Project_ChenPeter_JaneChua.zip

5.2 ITEM B: VIDEO SUBMISSION

- Using text or narration, describe and demonstrate how your Vivado project works in the video.
  - For each feature, remember to let me know …
    - What are the switches/pushbuttons to activate this feature?
    - How does the feature work?
  - If you are developing something on the OLED, do your best to make sure that the OLED screen is captured clearly in video.
- Name your file as Project_<Student_1_Name>_<Student_2_Name>.mp4 Eg. Project_ChenPeter_JaneChua.mp4

6. OTHER USEFUL RESOURCES

6.1 PICTURE2PIXEL

- Picture2Pixel is an open-source Python library designed to transform images into pixel art that can be displayed on FPGA-driven OLED screens. By employing advanced computer graphics techniques, this library preprocesses images to minimize distortion during the pixelation process.
- https://www.comp.nus.edu.sg/~guoyi/project/picture2pixel/