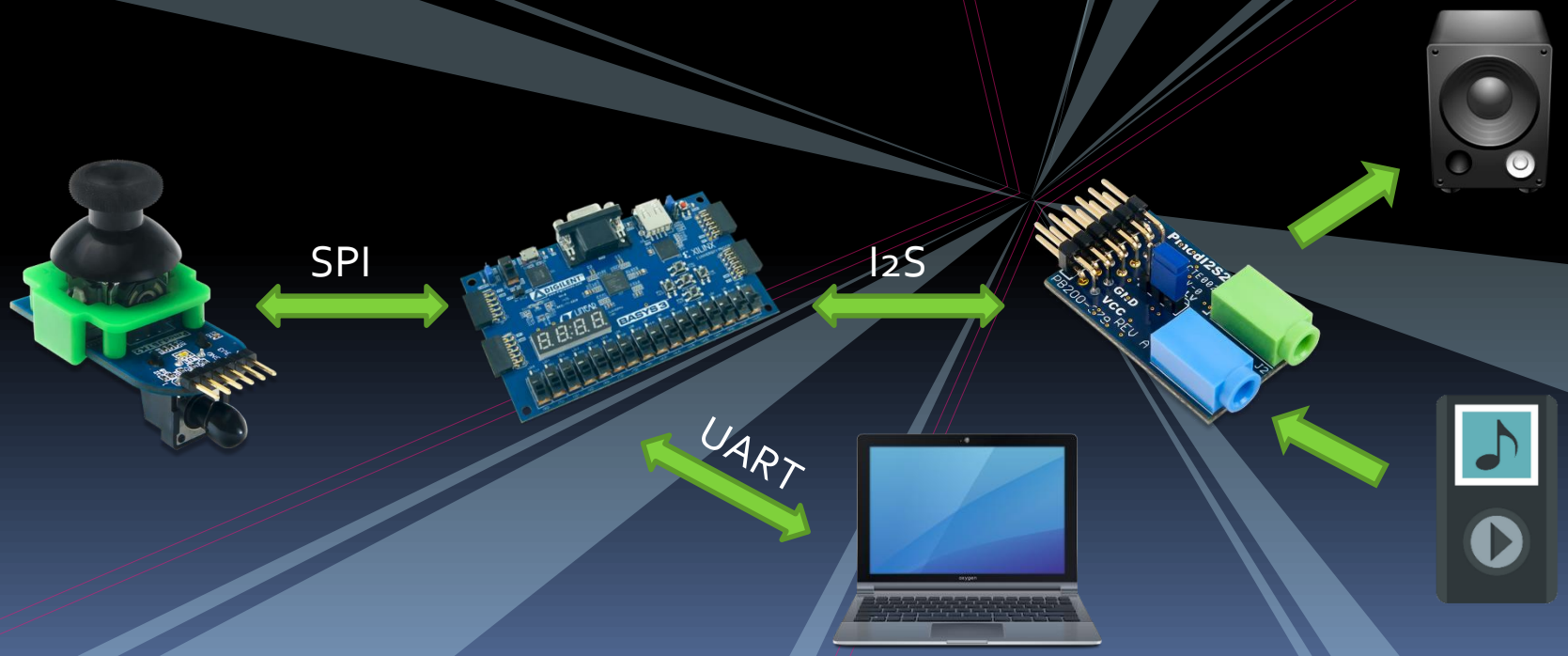
PW OVERVIEW

PROJECTS

- Mixer
- ColorToGrayScale

Project Goal

The objective is to build a "Digital Audio Console" by using the Pmod_I2S2, Pmod_JSTK2 modules and IP-Cores

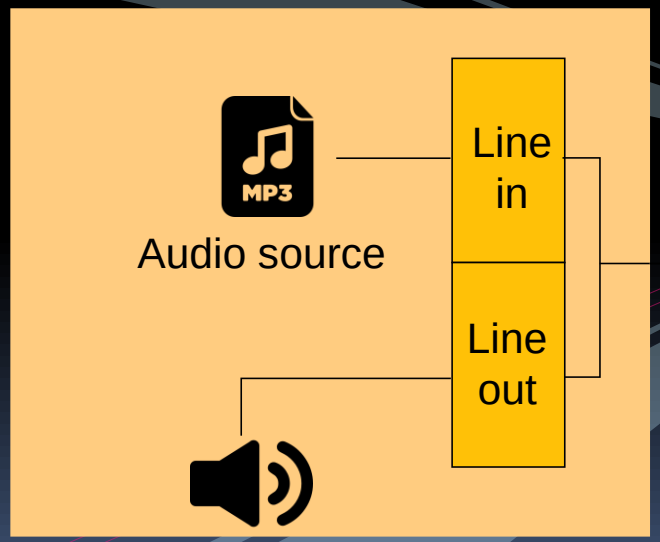


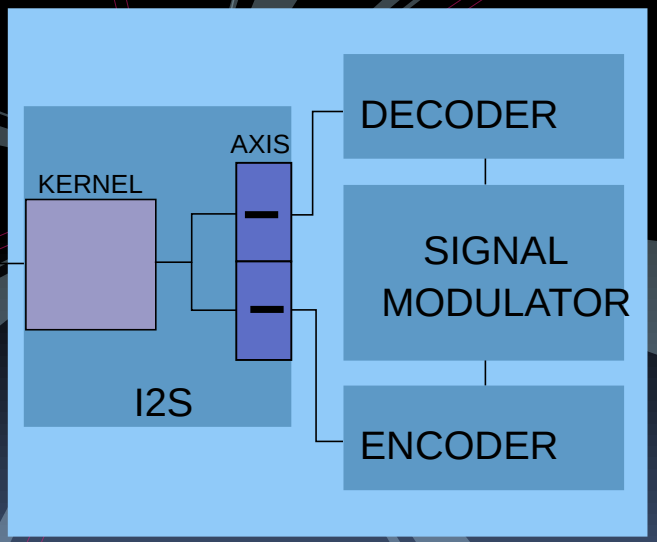
DEVICES

- Codec
- Joystick

Codec

Audio Source Sampling, I2S "Implementation", Encoder/Decoder, Signal Manipulation

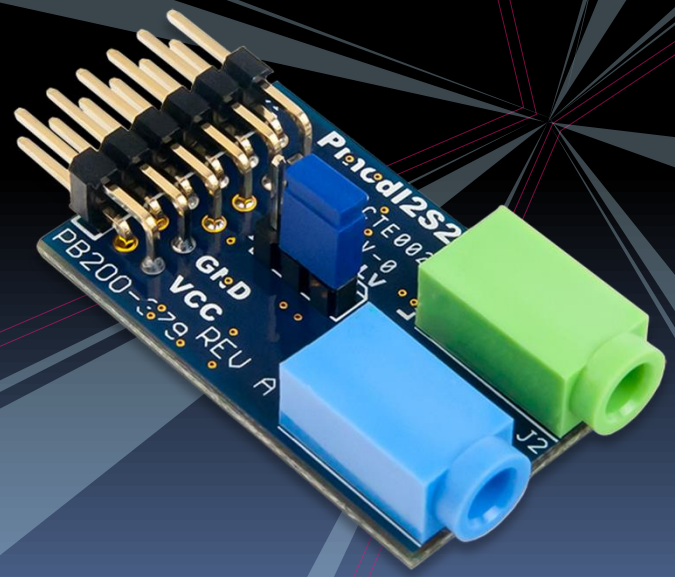




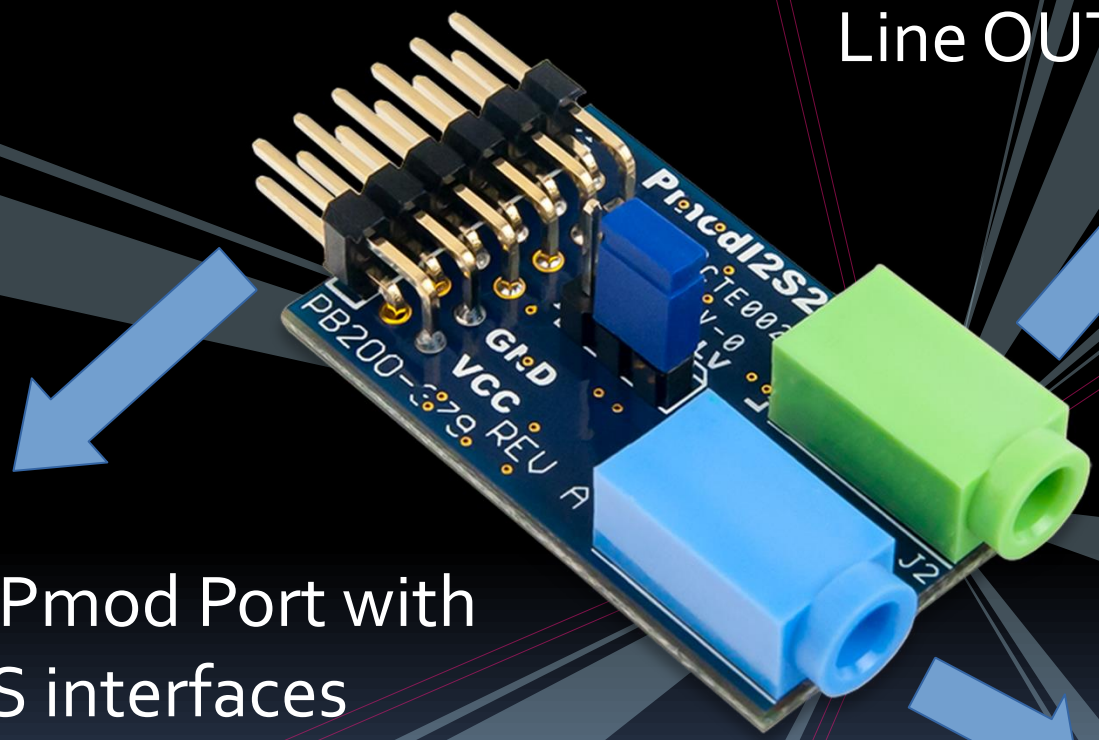
Pmod I2S2 module

At the beginning of LAB3, we will give each one of you a Digilent Pmod I2S2 module.

This can be connected to your Basys3 board through the Pmod connectors.



Digilent Pmod I2S2 components



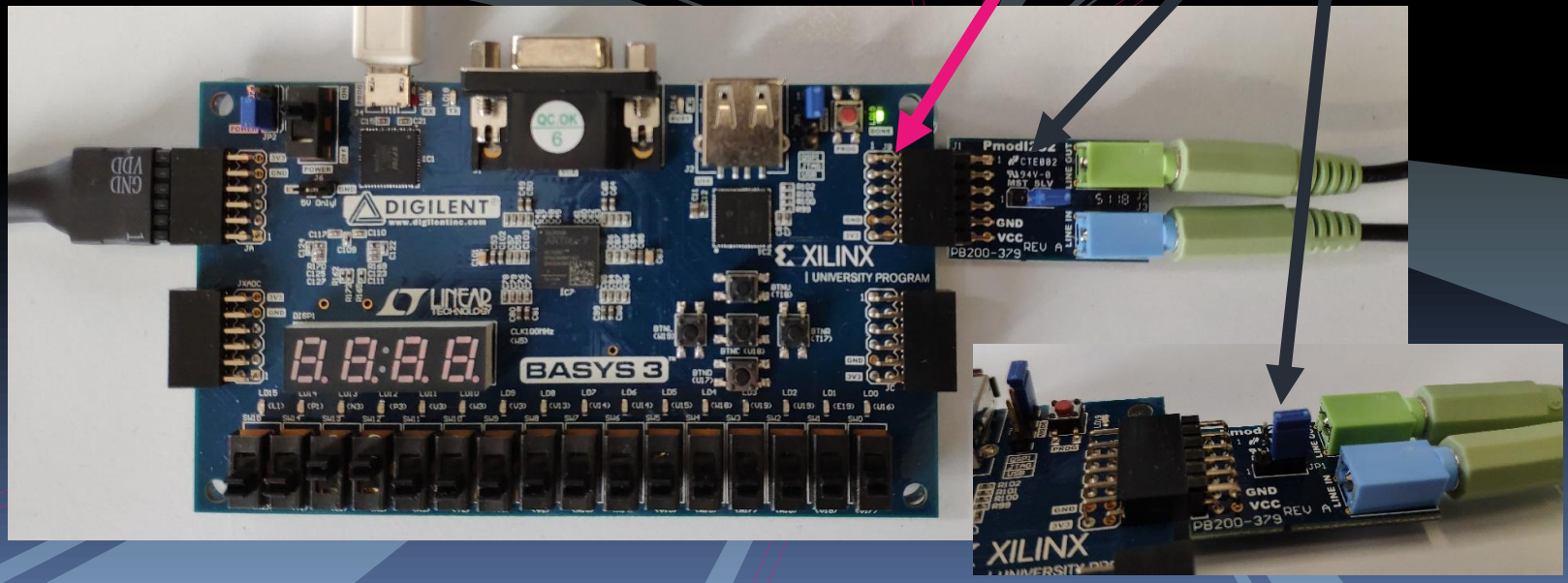
Line OUT (24 bit D/A)

12-pin Pmod Port with
two I2S interfaces

Line IN (24 bit A/D)

How to connect the I2S2 module

The provided constraints are for the JB connector (top right of the board). Also make sure that the jumper on the PmodI2S2 module is on the **SLV** position (right position).



Pmod I2S2 interfacing

The Digilent Pmod I2S2 module protocol is fully described in its reference manual.

In short, it uses the I2S protocol for both receiving an input audio signal from a source (through an ADC) and sending back an audio signal by means of a DAC to any kind of "speaker".

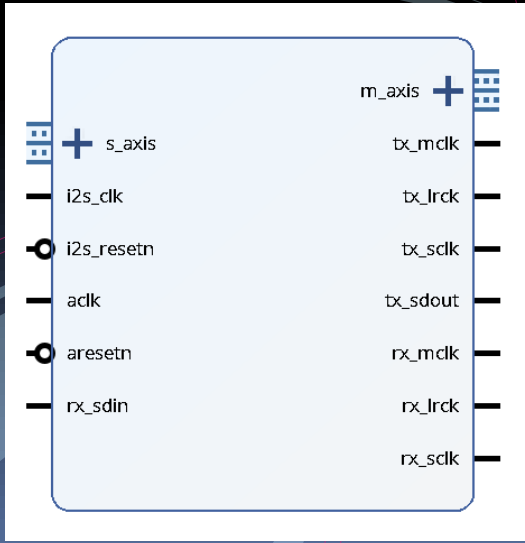
Audio Signal Format

- Typical sound frequency passband of the ears is between 20 and 20KHz.
  (Shannon theorem => 44.1 Ksamples)
- For a good quality audio we choose a 24bits/channel depth.
  (we consider stereo audio: L/R channels)

I2S IP-Core

The Inter-IC Sound (I2S) is a popular serial protocol for digital audio devices connections.

To ease your work, we will give you a "AXI4-Stream I2S2" IP-Core, similar to the SPI and UART ones.



I2S IP-Core

The provided Pmod-I2S2 IP-Core has:

- Two I2S interfaces, to be connected to the external pins of the FPGA.
- Two AXI4-Stream interfaces, to read the data from the ADC or to send the data to the DAC.
- Two clocks:
  - one for the I2S and (22.579 MHz)
  - one for the AXI4-Stream (100 MHz)
- Two input active-low reset signals (one for each clock domain).

I2S IP-Core: 44.1 kHz Audio

The IP core needs a 22.579 MHz clock (to be connected to i2s_clk) and a 100 MHz clock (to be connected to aclk input).

All the other ports have to be made external, and assigned to their correct pins.

Pmod-I2S2 AXI4-Stream format

The AXI4-Stream interface has an additional line called **TLAST**.

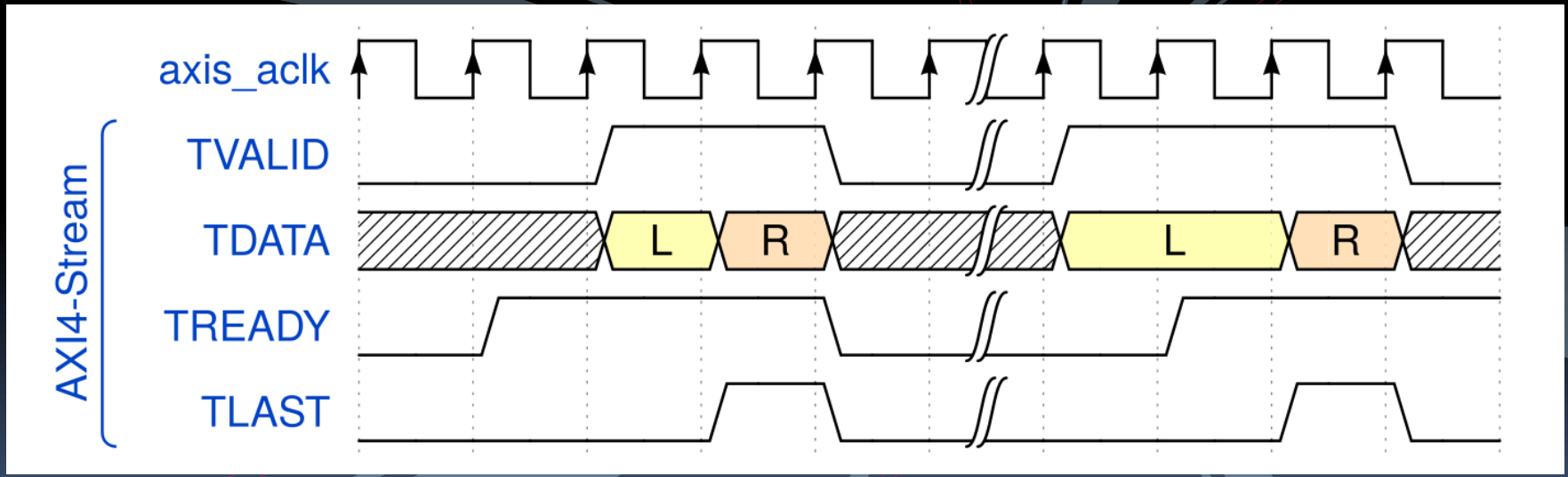
Each packet is composed by two 24-bits words: the first one is the audio data of the left channel, the second one the audio data of the right one.

TLAST is asserted on the second word; in other words:

- TLAST = 0: left channel
- TLAST = 1: right channel

Pmod-I2S2 AXI4-Stream format

In this example, two «packets» have been transferred, first the left channel and then the right one for each one.



DEVICES

- Codec
- Joystick

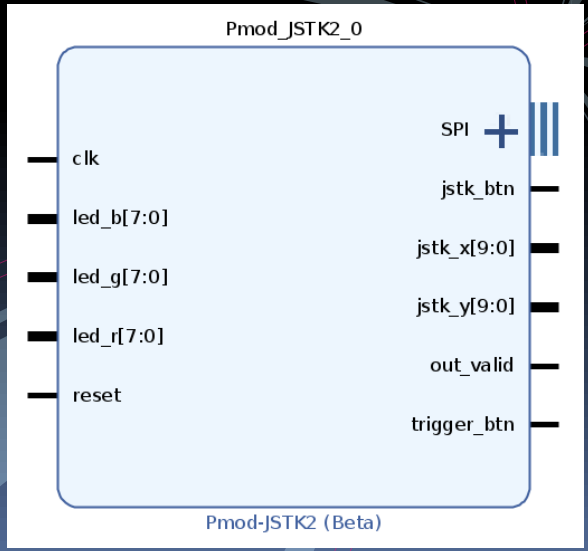
Pmod-JSTK2

The Digilent Pmod JSTK2 is a versatile user input device that can be easily incorporated into a wide variety of projects. It has a two-axis joystick on a center button, a trigger button, and a programmable RGB LED capable of 24-bit color.

Pmod-JSTK2 IP-Core

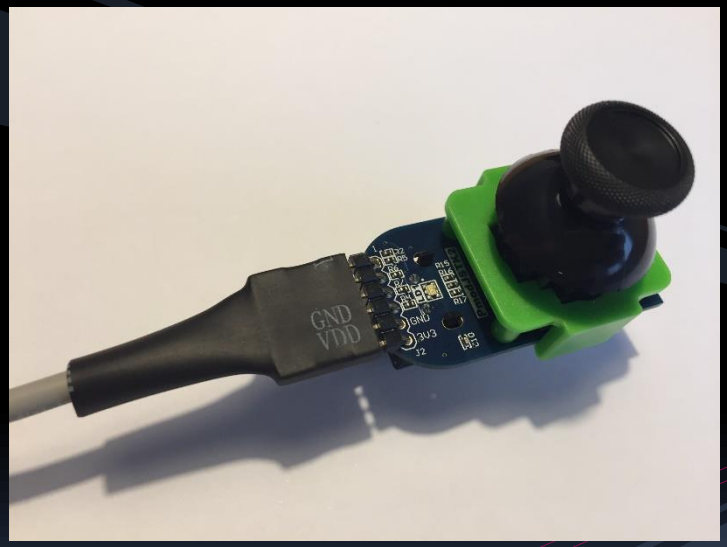
The Serial Peripheral Interface (SPI) is a popular serial protocol for digital devices connections.

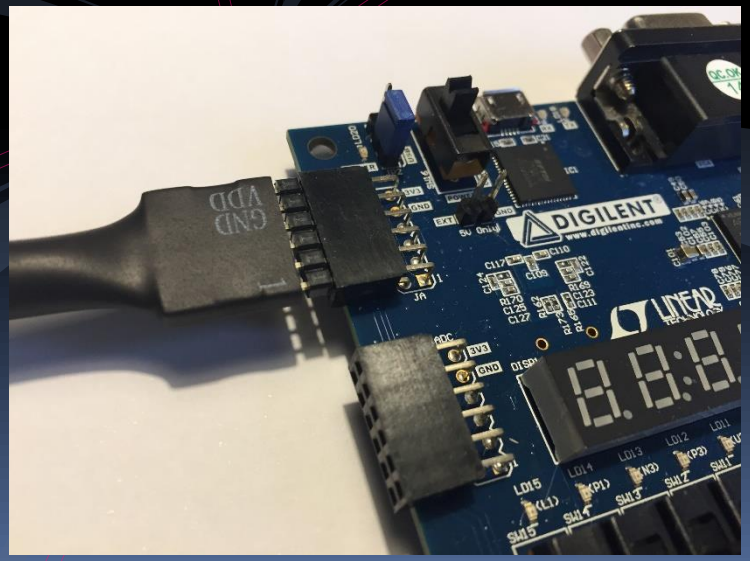
To ease your work, we will give you a "Pmod-JSTK2" IP-Core.



How to connect the Joystick

Connect the Joystick with the provided cable paying attention to the VCC and GND position

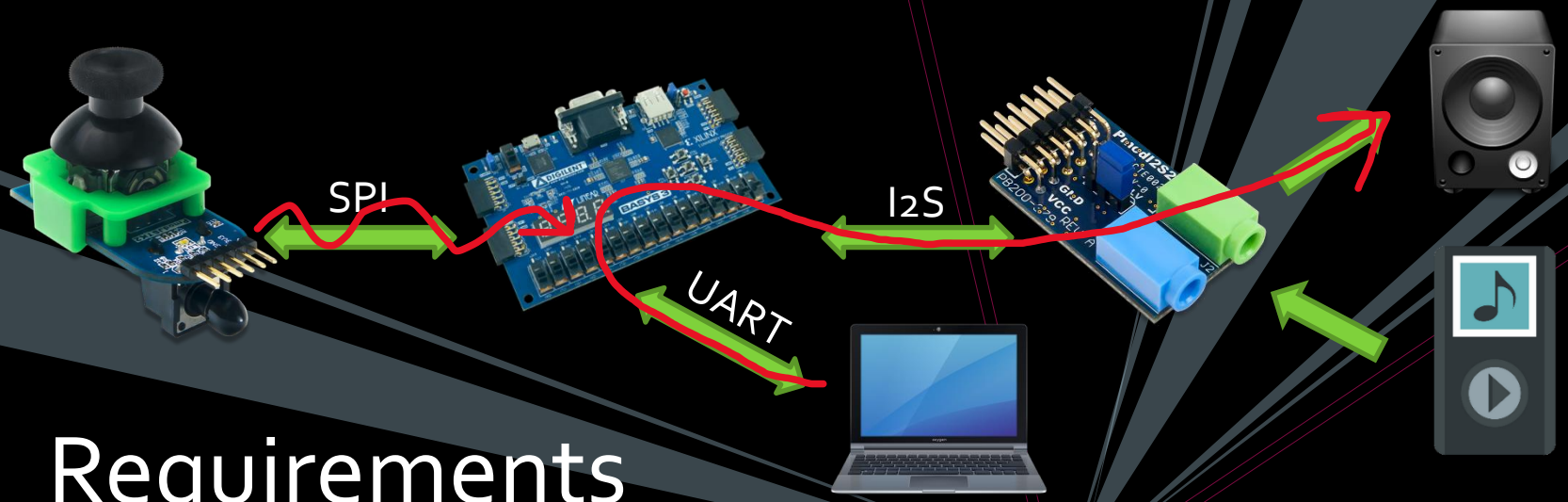




MIXER

- Step 1
- Step 2

First Step: Audio from UART



Requirements

The output audio (I2S output) should reproduce the input one coming from the UART with some "effects" applied:

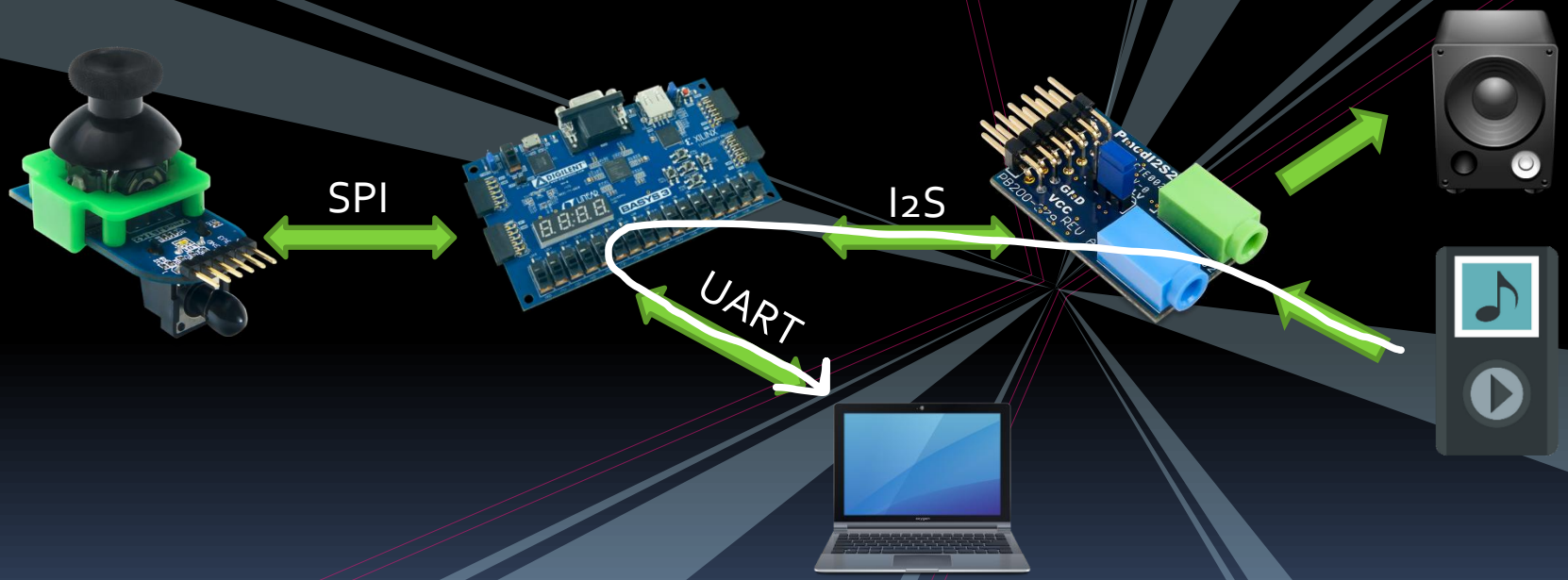
- The vertical axis of the Joystick should control the volume of the output audio.
- The horizontal axis should control the audio balance.

We must be able to see the spectrum of this audio on the UARTPlayerGUI.

First Step: Audio from UART

Besides, we must be able to see also the spectrum of the audio coming from the I2S input, on the UARTPlayerGUI.

This signal is not affected by the joystick.



First Step: Audio from UART

- Pushing* the "trigger button" mutes or unmutes the output channel.
- Pushing* the "joystick button" enables or disables a moving average filter (depth=32).
- The LED on the PmodJSTK2 module should show the status: muted (red), filter active (blue), no effects (green).

* toggles the status, not just "active when pressed"

Audio Signal Format (1/2)

- Typical sound frequency passband of the ears is between 20 and 20KHz. (Shannon theorem => 44Ksamples)
- For a good quality audio we choose a 16bits channel depth.
  (we consider 2 audio channel: L/R)

Audio Signal Format (2/2)

CHANNELS bits Nº samples/s

Which BAUD_RATE is necessary to sustain this bitrate?

BAUD_RATE: 2Mbit

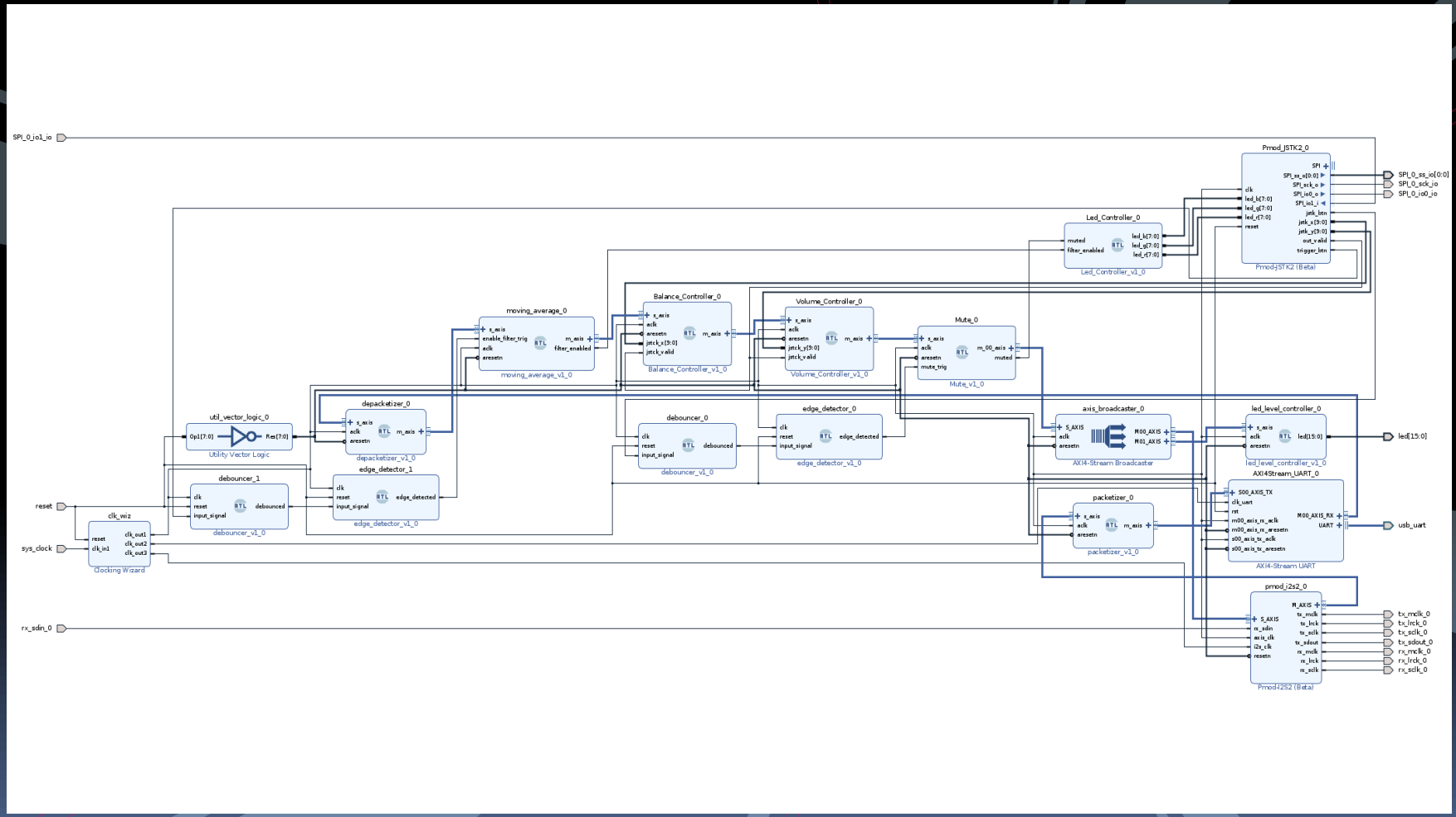
RS 232 - UART

UART protocol commonly uses an oversampling of 16 times the BAUD_RATE.

BAUD_RATE

To avoid UART errors, connect clk_UART to a clock with a frequency multiple of 32 MHz.

Block Design – Step One (1/2)

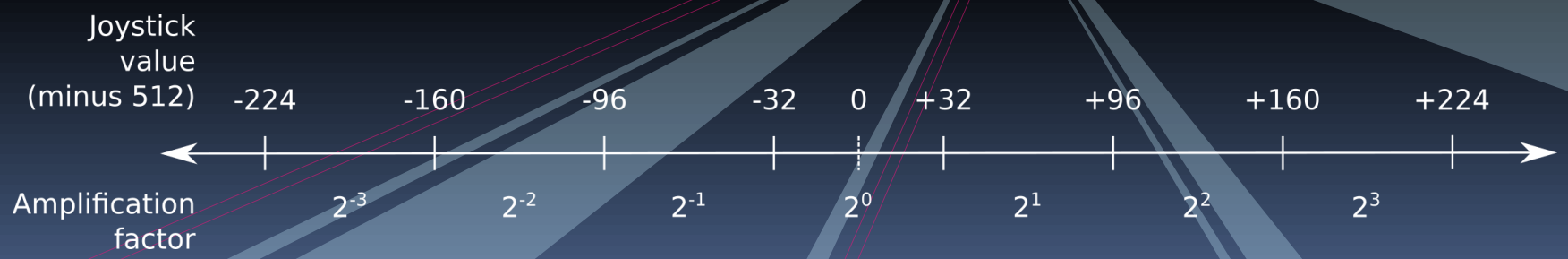


Block Design – Step One (2/2)

- Volume Controller, Balance Controller, Moving Average, Led Controller (joystick), Edge Detector and Debouncer are already provided.
- You have to implement the following remaining modules: Packetizer, Depacketizer, Led Level Controller.

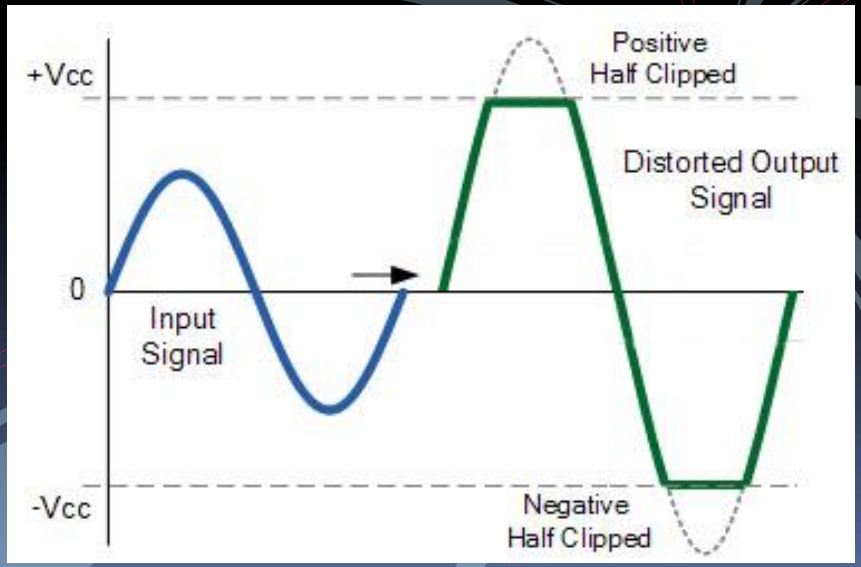
Details: volume (1/2)

We perceive “loudness” in a logarithmic way, so the volume control should be exponential. The amplification factor should double every 2^N “joystick units” (with the center in the half of the joystick dynamic, with N as generic). N=6 returns good results.



Details: volume (2/2)

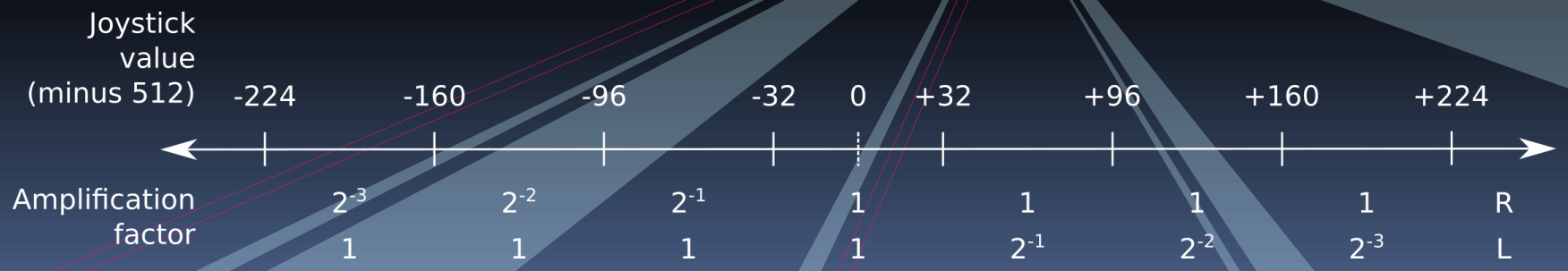
Be careful when multiplying: the signal must saturate at the maximum possible value ("clipping"), you must handle this manually to avoid unexpected results.



Details: balance control

Use an exponential amplification factor also for the balance control.

- Moving the joystick to the right decreases the left channel volume.
- Moving the joystick to the left decreases the right channel volume.



Details: mute

It is quite self explanatory

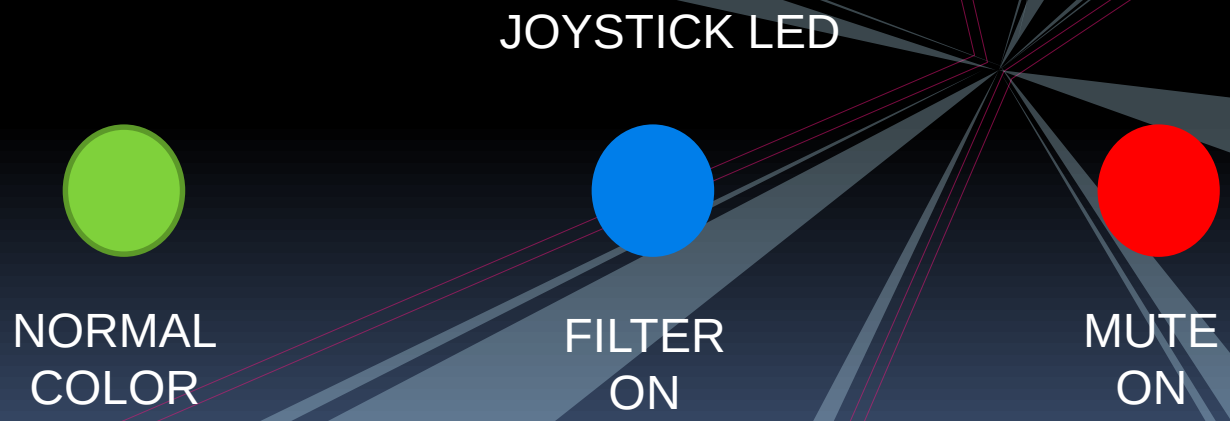
Details: moving average filter

The moving average module must be able to selectively apply a moving average filtering (of a fixed order of 32, set by generic) on the samples.

The module should filter the samples when “enable_filter” is high, and should simply pass the samples unmodified when “enable_filter” is low.

Details: LED controller (joystick)

LED controller: sets the Pmod-JSTK2 LEDs to the correct colors, depending on its inputs ("mute_on" and "filter_on")



Details: LED level controller

LED level controller: it turns on the 16 LEDs on the board, depending on the level of the audio at the output (right and left channels are averaged).

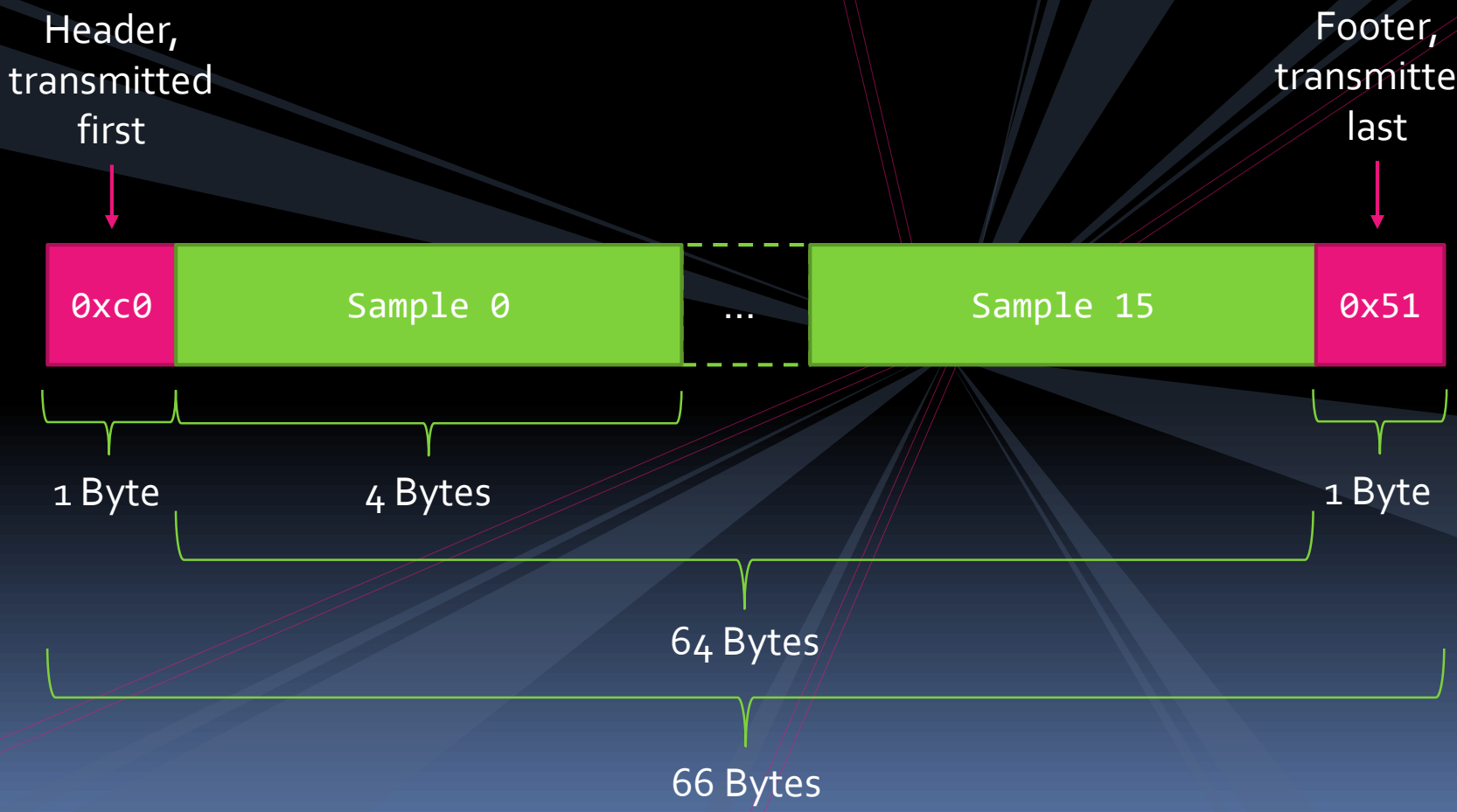
Details: Packetizer and Depacketizer 1/5

Using the improved AXI4-Stream UART module, build a packetizer and a depacketizer to reliably transmit data between the PC and the FPGA:

- depacketizer: decode packetized data (66 bytes, see next slide) to «raw» audio data (16 bits per channel, with TLAST high on every right channel sample
- packetizer: modified audio data (16 bits per channel, TLAST) to packetized data (66 bytes, see next slide)

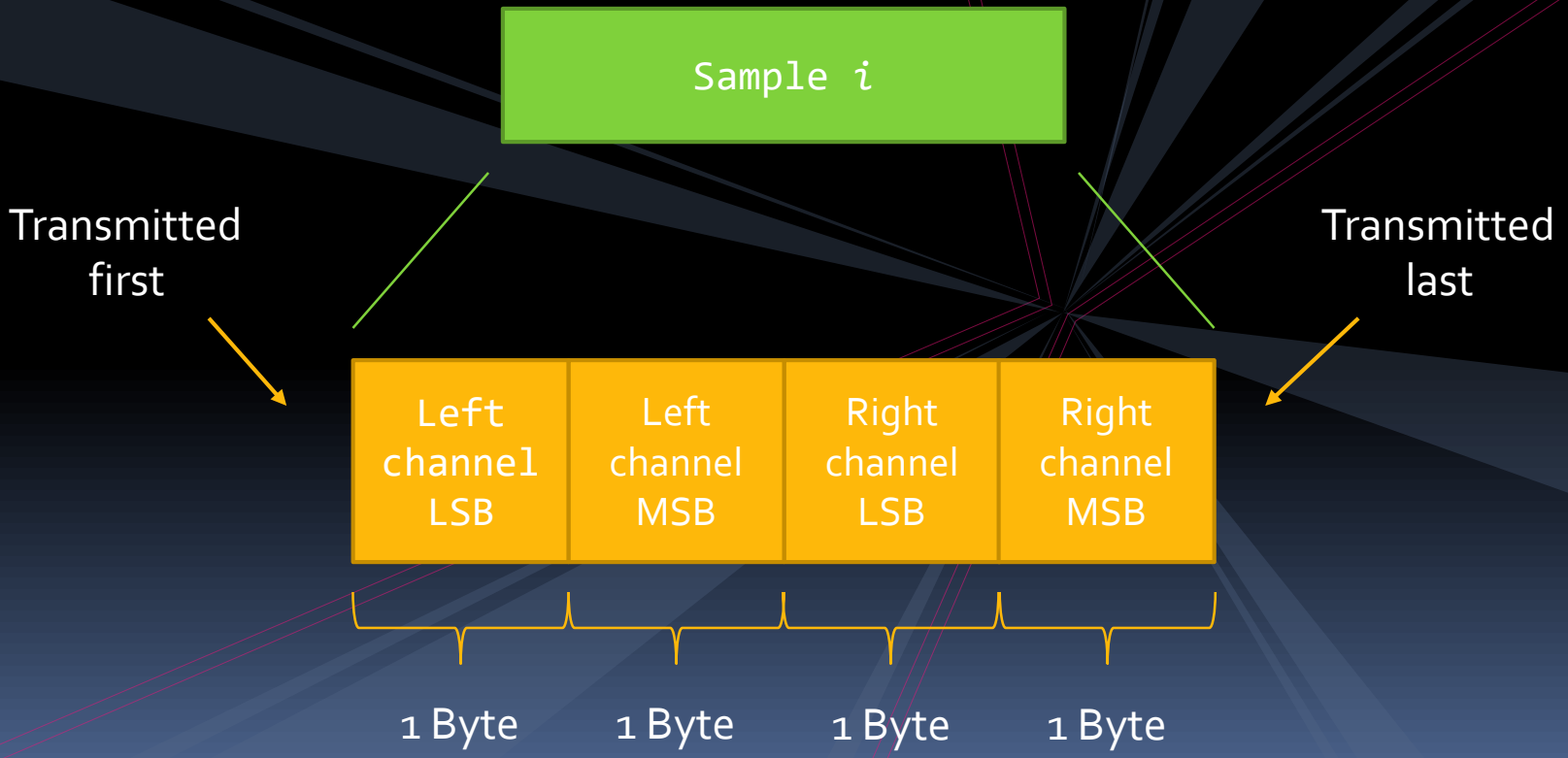
Details: Packetizer and Depacketizer 2/5

The packet format is the following one:

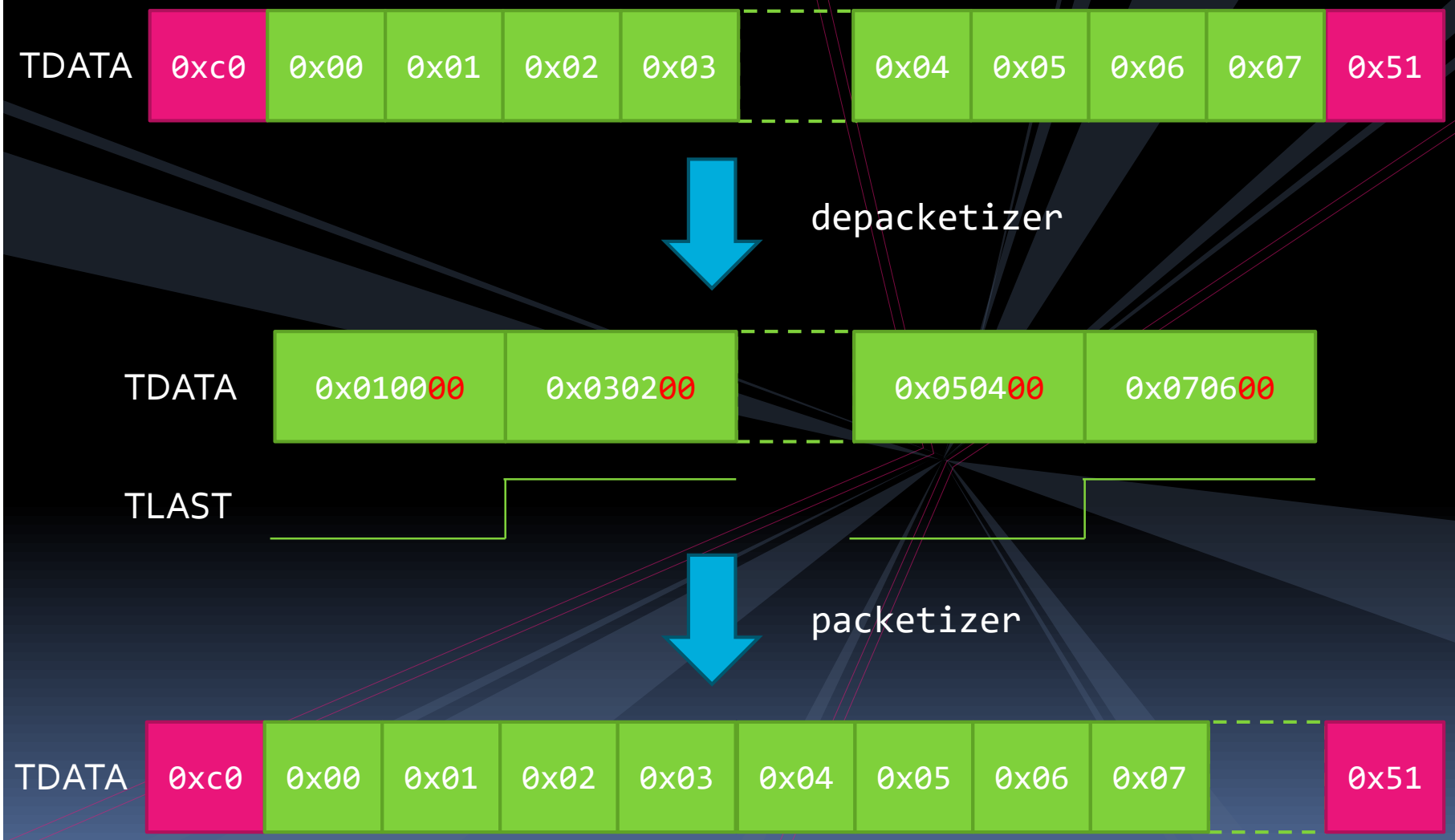


Details: Packetizer and Depacketizer 3/5

The packet format is the following one:



Details: Packetizer and Depacketizer 4/5



Details: Packetizer and Depacketizer 5/5

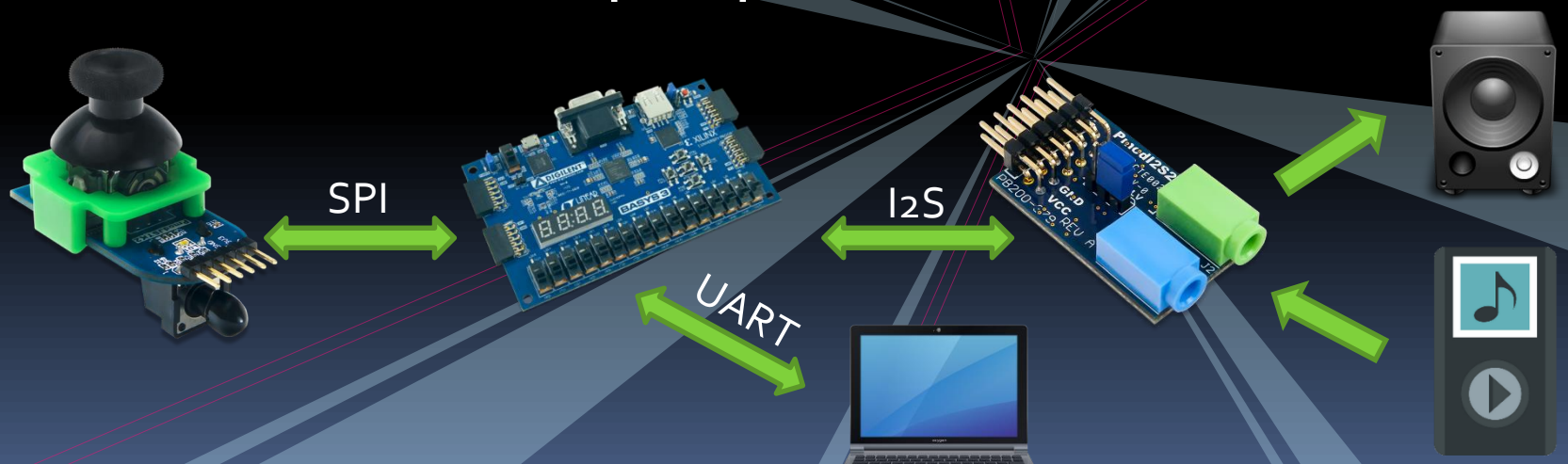
The zeroes in red in the depacketized data (previous slide) is an 8-bit zero-padding. This is necessary to have width-matching in all the audio-processing modules, which have 24-bit data-width. Indeed, the destination of the dataflow is the I2S2 module, having 24-bit data-width.

MIXER

- Step 1
- Step 2

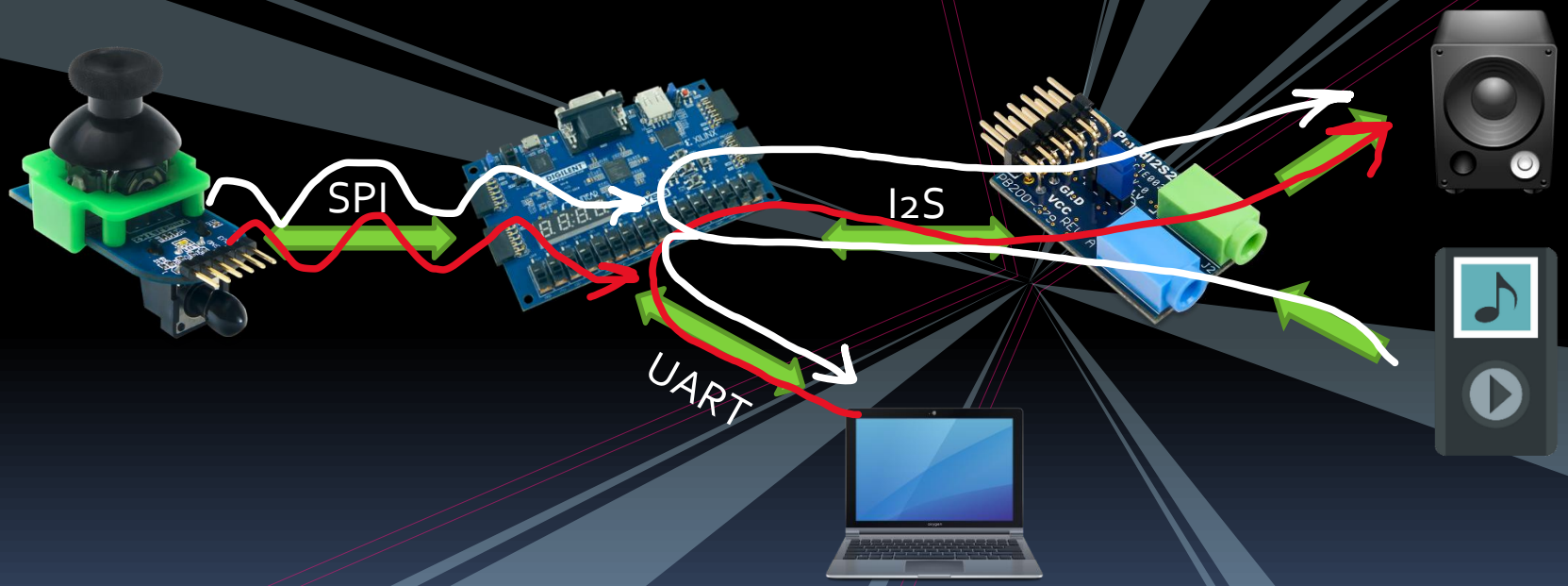
Second Step: Mixer (1/3)

The goal is to mix the two audio channels (channel 0 is the audio source coming from the I2S module, channel 1 is the one coming from the UART interface), so to hear both of them from the Pmod I2S2 output port.



Second Step: Mixer (2/3)

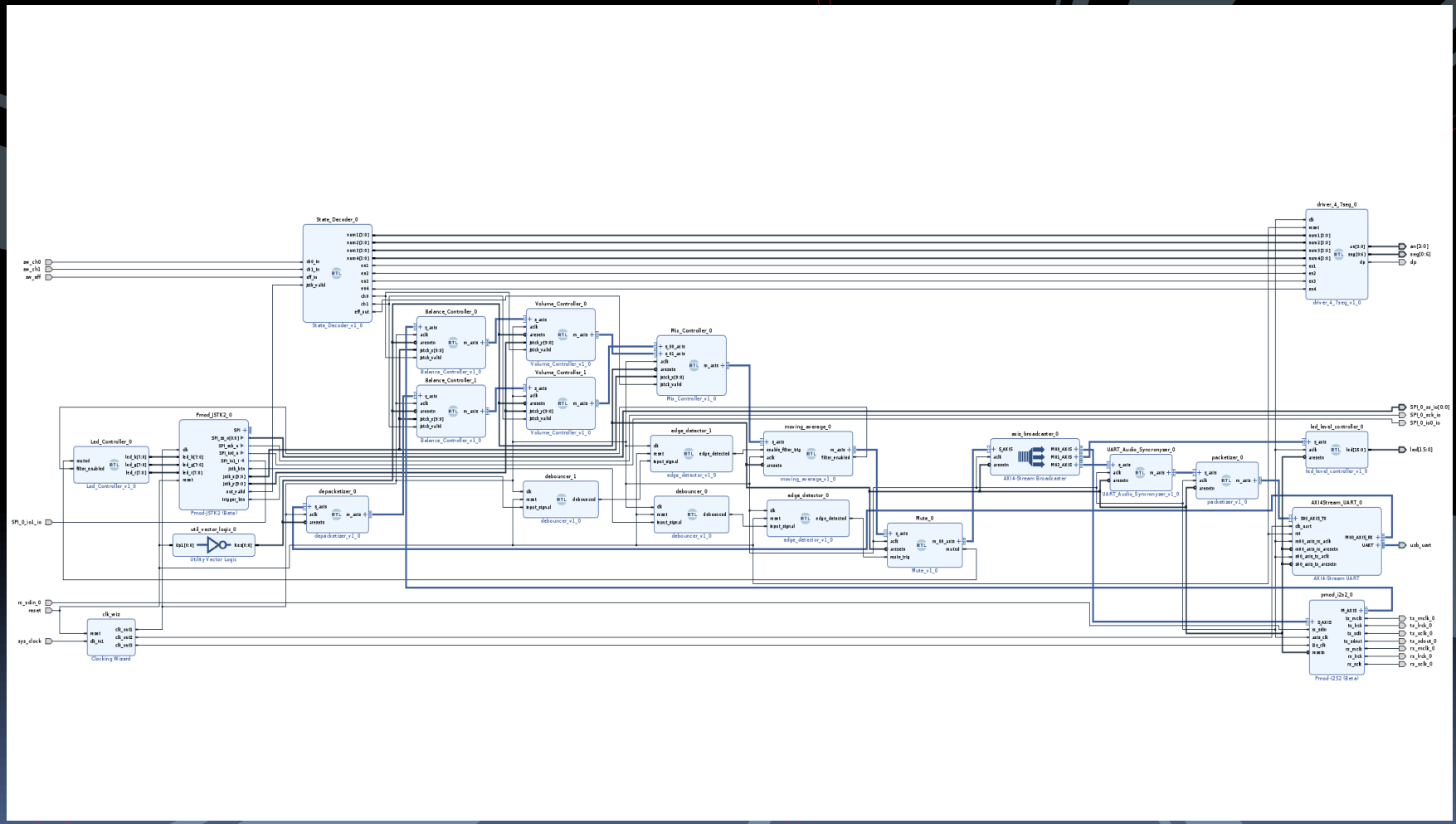
Besides, we must be able to see also the spectrum of the mixed audio on the UARTPlayerGUI.



Second Step: Mixer (3/3)

- Provided module: UART_Audio_Synchronizer: this is needed to synchronize the two audio sources, in order to avoid problems in the audio quality.
- You have to write the following three modules: Mix_Controller, State_Decoder, Driver_4_7segment.

Block Design – Step Two (1/2)



Block Design – Step Two (2/2)

- Volume Controller and Balance Controller are now duplicated, so that each channel has his own control over these parameters.
- Moving average and Mute are instead global effects.

Details: Mix Controller (1/2)

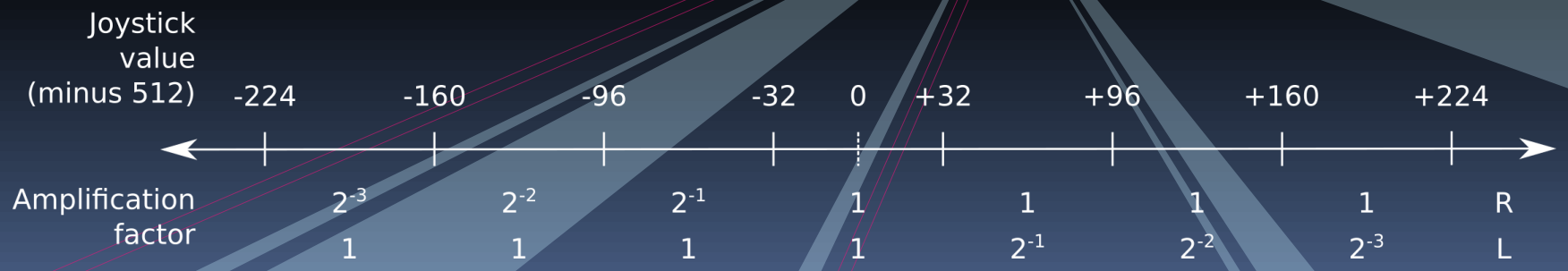
This is the module which manages the sum of the two audio data coming from each one of the two channels, allowing two hear both audio sources.

Just like a crossfader for DJ mixers.

Details: Mix Controller (2/2)

It also has to include a balance control effect (cross-fader) working in this way:

- Moving the joystick to the right lets the channel 0 (I2S) fade-out.
- Moving the joystick to the left lets the channel 1 (UART) fade-out.



Details: State Decoder (1/3)

The task of this module is twofold:

- It has to control the activation/deactivation of the Mixer module (see previous slides).
- It has to send its state (in nibbles) to the 7-segment display driver (will be described later).

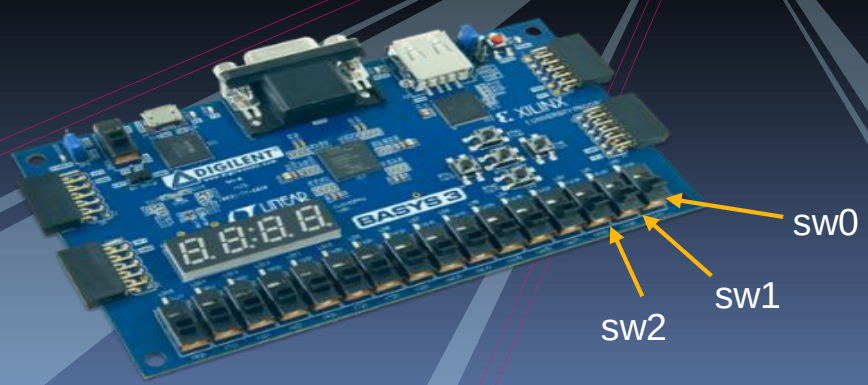
Details: State Decoder (2/3)

Let's see in more details how it works:

- By activating sw0 of the Basys3, it activates the Mix Controller and provide the Crossfade mode described in the previous slides, and has to print "[]EFF" on the 7-segment display (sending the correct hex code to it). ([] means nothing is printed in that spot).
- By activating sw1, the system focuses just on channelo (I2S audio track) and allow to control its volume and its balance (as it was in step one). (sw0 must be turned off). On the 7-segment display has to be printed "[][]C0".

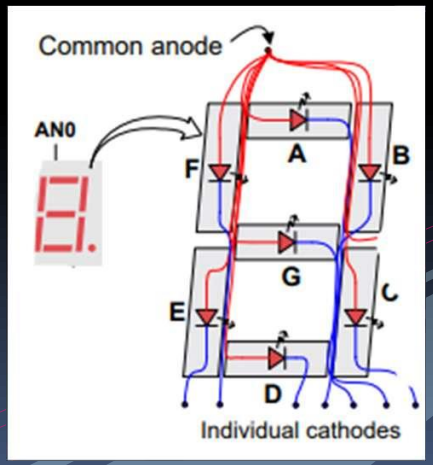
Details: State Decoder (3/3)

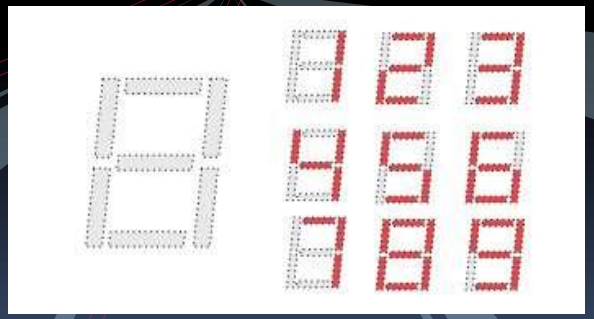
- By activating sw2 of the Basys3, the system just controls channel1 (UART audio track), controlling its volume and balance (as it was in step one). On the 7-segment must be displayed "C1[][]". (sw0 must be turned off in this mode).
- By activating both sw1 and sw2, the system can control balance and volume of both tracks together, and on the 7-segment is displayed "C1C0".



Details: 7-Segment Display (1/6)

A 7-Segment display is composed by 7 LEDs with a common Anode for the 7 LEDs and 7 individual Cathodes (1-per-LED)

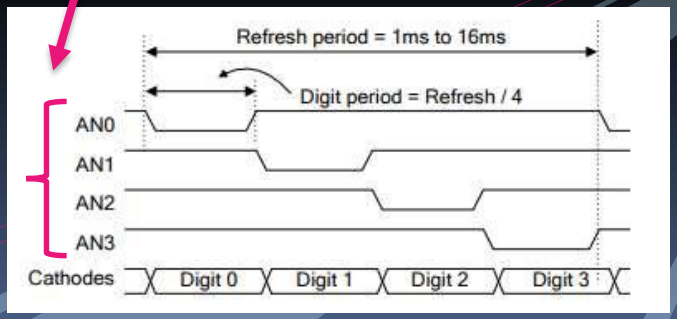


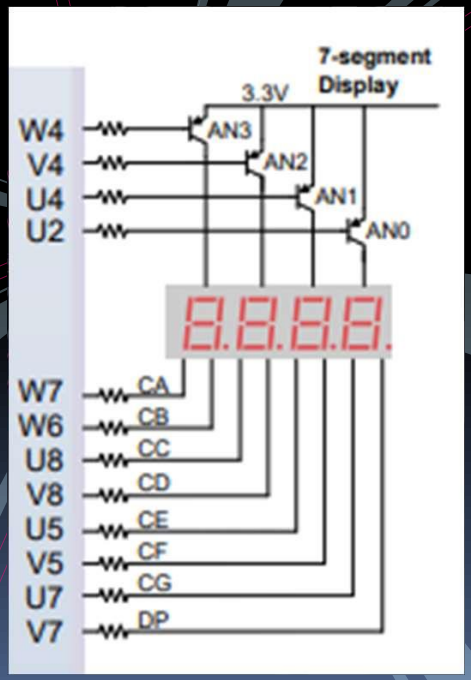


Details: 7-Segment Display (2/6)

If we have more than one display, we have to multiplex the Digits and the selection to minimize the Outputs.

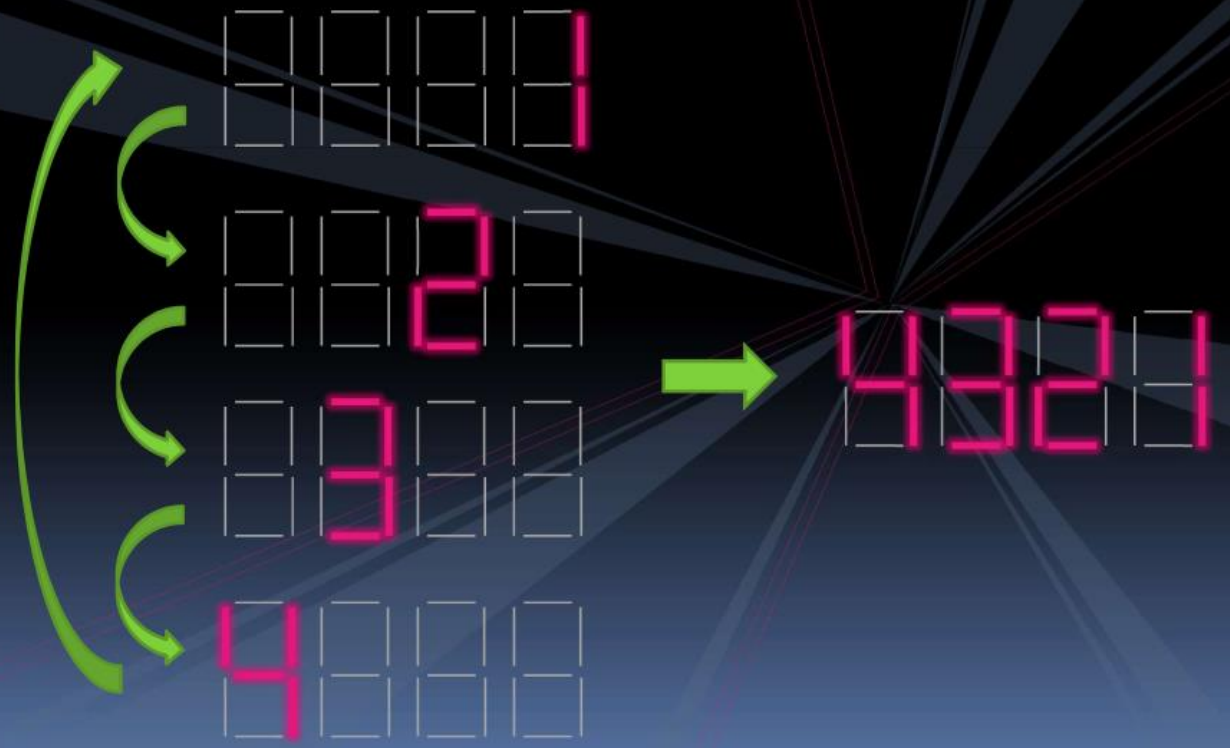
Warning: "enabled" when LOW





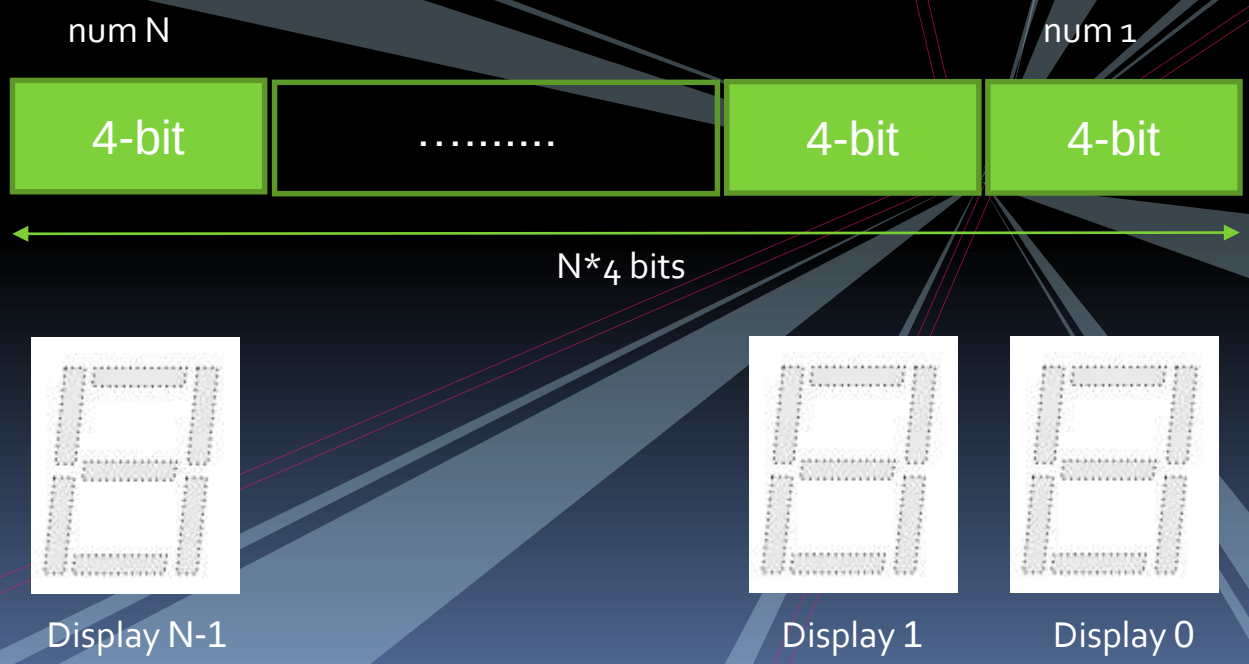
Details: 7-Segment Display (3/6)

If we multiplex the digits fast enough, they appear all turned on.



Details: 7-Segment Display (4/6)

The nibbles coming from the State_Decoder module are multiplexed one-by-one on the 7-Segments Display.



Details: 7-Segment Display (5/6)

```
entity driver_4_7seg is
    Generic(
        mux_time_ms: positive:=1;
        clock_period_ns: positive :=10
    );
    Port (
        clk : in STD_LOGIC;
        reset: in STD_LOGIC;
        num1, num2, num3, num4: in STD_LOGIC_VECTOR(3 downto 0);
        en1, en2, en3, en4: in STD_LOGIC;
        an: out STD_LOGIC_VECTOR(3 downto 0);
        seg: out STD_LOGIC_VECTOR(0 to 6);
        dp: out STD_LOGIC
        );
end driver_4_7seg;
```

Details: 7-Segment Display (6/6)

```
with num select seg_s <=
"0000001" when "0000",
"1001111" when "0001",
"0010010" when "0010",
"0000110" when "0011",
"1001100" when "0100",
"0100100" when "0101",
"0100000" when "0110",
"0001111" when "0111",
"0000000" when "1000",
"0000100" when "1001",
"0001000" when "1010",
"1100000" when "1011",
"0110001" when "1100",
"1000010" when "1101",
"0110000" when "1110",
"0111000" when "1111",
"1111111" when others;
```