VLC Simulator

Overall description of the VLC simulator.

# Main

The Main class is the one responsible for wrapping up all other classes’ objects. It can be called with the following arguments:

* ASIC:
  + Sets to use an integrated circuit modelling of choice.
* Simulator:
  + Chooses between Cadence and Tanner integrated circuit simulators.
* Modulation:
  + What modulation to be used, OFDM, OOK, etc.
* …

The system starts with the desired information to be transmitted, handled by class Message(), which converts the input into a stream of bits. That info is then mapped by Mapping(), given the input bit stream, which converts to a parallel stream, which is modulated by the Modulator(),that creates a digital signal to be transmitted. That signal passes through a DAC(), from where the signal goes to the Transmitter(). The transmitted signal drives a LightSource(), that can be a LED, for instance. The light signal passes through the Channel(), and provides the CIR (Channel Input Response), that is used inside the Receiver(). In the receiver side, the light signal convoluted with the channel response is then transduced by the Detector(), that generates a photocurrent. That current is provided to the ROIC(), that converts it to a voltage output. This circuit may be one of the supported Simulators(). Next, that voltage is converted to a digital representation by a ADC(), which in turn is used by the Demodulator(), generating the final received bit stream value. Ultimately, a comparison between the transmitted and received info is analyzed by the MeritFunctions(), given for instance, the BER (Bit-Error-Rate).

# Message

This class expects an input information to be transmitted in the form of: “Image”; “List of strings”, etc.

It features a stream of bits as standard output. At the end, the received bitstream returns to Message for code back final message.

# Mapping

It receives a stream of bits from the Message with size NBITS, converts it from serial to parallel, and apply a mapping, such as QAM. It also performs the de-mapping.

# Modulator

It gets the numpy array of the mapped\_info, and apply the desired modulation type, such as OOK or OFDM. It also performs the demodulation operations.

# OFDM

From the mapped information, it generates the OFDM symbol for transmission, while applying IDFT and the cyclic prefix. Also do the opposite direction, as well, while on the receiver end.

# DAC

It converts the input tx\_data, from the modulator, and applies the DAC operation. This can be done through a circuit simulation, or by mathematical rounding operation. There is also an option to bypass this stage, if required.

# Transmitter

This class is a wrapper for the transformation of the analog signal into optical power. It uses a LightSource object to call do the calculations for what is the transmitted optical power profile. Can instantiate different “lamps”, where each lamp is an array of LightSource.

# LightSource

Contains all info regarding a LightSource type.

# Channel

Gets the data from the Transmitter, and apply the channel impulse response (CIR) to input data. The CIR can be calculated by RayTracing, if desired.

# Receiver

This class is a wrapper for the transformation of the optical power into an analog signal. It uses a Detector object to do the calculations for what is the final photocurrent profile. Can instantiate different arrays of Detectors.

# Detector

Contains all info regarding a Detector type.

# ROIC

Contains all info regarding a circuit to be simulated, if not bypassing it, with circuit\_simulation=False.

# BouncingPixel

Inherited from ROIC, it contains the complementary functions to make correct analysis of that circuit.

# Simulator

Basically, a ‘class wrapper’, to provide common methods for other classes.

# Virtuoso

Initializes the netlist for Virtuoso, and call the specified simulator. Return data are the waves.

# ADC

It converts the voltage data, after the ROIC, into a digital representation. This can be done through a circuit simulation, or by mathematical rounding operation. There is also an option to bypass this stage, if required. This data goes back to the demodulator.

# MeritFunctions

Implements the functions to access the performance of the system, like the Bit-Error-Rate (BER), SNR, etc.