Testcases documentation

Unit Test: TestConfig

Overview

The TestConfig class tests the functionality of the Config class, which loads configuration settings from a specified file. This unit test ensures that the configuration values are correctly parsed and assigned.

Key Components

- **setUp()**:
    - Creates a sample configuration file (sample_config.cfg) with predefined values to be used during testing.
- **tearDown()**:
    - Cleans up by deleting the sample configuration file after each test to maintain a clean test environment.
- **test_load_config()**:
    - Validates that the load_config method correctly reads the configuration values from the sample file.
    - Checks that attributes such as stream_duration_ms, burst_size, and others match the expected values.

Purpose

This unit test ensures that the Config class can successfully load and correctly assign the configuration settings necessary for packet generation, verifying the integrity of the configuration management process.

Unit Test: TestECPRI

Overview

The TestECPRI class tests the functionality of the eCPRIFrame class, which is responsible for constructing eCPRI frames. The tests verify that frames are created correctly based on the number of IQ samples specified.

Key Components

- **test_create_ecpri_frame()**:
    - Tests the creation of an eCPRI frame with a specified number of IQ samples (4 in this case).
    - Asserts that the frame is a bytearray and checks that the total length is as expected (16 bytes for IQ samples plus 4 bytes for the header).
- **test_create_ecpri_frame_large()**:
    - Tests the creation of an eCPRI frame with a larger number of IQ samples (80).
    - Similar checks as in the previous test, ensuring that the frame is a bytearray and the total length matches the expected size.
- **test_create_header()**:
    - Verifies that the eCPRI header is created correctly.
    - Asserts that the header is a bytearray, checks its length (4 bytes), and verifies that the first byte is the expected protocol version (set to 0x00 for IQ Message Type).

Purpose

These unit tests ensure that the eCPRIFrame class constructs frames and headers correctly, confirming the integrity of eCPRI message generation in the packet generation system.

Unit Test: TestPacketGenerator

Overview

The TestPacketGenerator class is designed to test the functionality of the PacketGenerator class, which handles the generation of Ethernet and eCPRI packets. These tests validate that the packet creation and associated methods work correctly based on the configuration parameters.

Key Components

- **test_create_ethernet_packet()**:
    - Tests the creation of an Ethernet packet.
    - Asserts that the generated packet is of type bytes after calling the _create_packet() method.
- **test_create_ecpri_packet()**:
    - Tests the creation of an eCPRI packet.
    - Similar to the previous test, it checks that the generated eCPRI packet is also of type bytes.
- **test_get_ether_type()**:
    - Validates the functionality of the _get_ether_type() method in the PacketGenerator class.
    - Tests that the correct EtherType is returned based on the specified packet type (ECPRI or ETHERNET).
- **test_generate_payload()**:
    - Tests the payload generation for Ethernet packets.
    - Checks that the fixed payload generated is equal to the expected byte string (b'THIS IS A TEST') when the payload type is set to FIXED.

Purpose

These unit tests ensure that the PacketGenerator class correctly generates Ethernet and eCPRI packets, manages EtherType retrieval, and produces the expected payloads. This verification is crucial for ensuring the integrity of packet generation in the Ethernet packet generation system.

Unit Test: TestUtils

Overview

The TestUtils class tests utility functions defined in the utils module, specifically focusing on CRC calculation and padding for Inter-Frame Gaps (IFGs). These tests ensure that the utility functions perform their intended functionality correctly.

Key Components

- **test_crc32()**:
    - Tests the crc32 function to verify the correctness of CRC calculation.
    - Asserts that the CRC for the input data b"Hello, world!" is equal to b'\xeb\xe6\xc6\xe6'.
    - Validates that the CRC for the input data b"THIS IS A TEST MESSAGE" is equal to b'\x19M\xdfI'.
- **test_get_ifg_padding()**:
    - Tests the get_ifg_padding function, which adds padding to ensure 4-byte alignment.
    - For a packet of 62 bytes (not a multiple of 4), asserts that the padding returned is b'\x07\x07'.
    - For a packet of 64 bytes (already a multiple of 4), asserts that no padding is needed, resulting in an empty byte string (b'').

Purpose

These unit tests ensure that the utility functions crc32 and get_ifg_padding work as expected. Validating the CRC calculation and the proper handling of padding is essential for maintaining data integrity and alignment in the packet generation process.

Additional Test Cases

Ethernet Packet Test Cases

1. **Regular Ethernet with Payload Size Less Than 46**:
    - **Description**: Tests the packet generation with a payload size smaller than the minimum Ethernet frame size.
    - **Expected Outcome**: The packet should be padded to meet the minimum frame size of 46 bytes.
2. **Regular Ethernet with Payload Size Greater Than 46**:
    - **Description**: Tests the packet generation with a payload size greater than the minimum Ethernet frame size.
    - **Expected Outcome**: No padding should be applied, and the packet should remain unchanged.
3. **Regular Ethernet with Payload Size Greater Than 1500**:
    - **Description**: Tests the packet generation with a payload size exceeding the typical maximum size for Ethernet frames.
    - **Expected Outcome**: The packet should be rejected, and IFGs should be sent instead.
4. **Regular Ethernet Exceeding Maximum Packet Size in Configuration**:
    - **Description**: Tests the packet generation with a total size exceeding the configured maximum packet size.
    - **Expected Outcome**: The packet should be rejected, and IFGs should be sent instead.

eCPRI Packet Test Cases

1. **eCPRI Packet with Payload Size Less Than 46**:
    - **Description**: Tests the eCPRI packet generation with a payload size smaller than the minimum Ethernet frame size.
    - **Expected Outcome**: The eCPRI packet should be padded to meet the minimum frame size.
2. **eCPRI Packet with Payload Size Greater Than 46**:
    - **Description**: Tests the eCPRI packet generation with a payload size greater than the minimum Ethernet frame size.
    - **Expected Outcome**: No padding should be applied, and the packet should remain unchanged.
3. **eCPRI Packet Exceeding 1500 Bytes**:
    - **Description**: Tests the eCPRI packet generation with a payload size exceeding the maximum size for Ethernet frames.
    - **Expected Outcome**: The packet should be rejected, and IFGs should be sent instead.

Purpose

These additional test cases ensure that both regular Ethernet packets and eCPRI packets are handled correctly based on their payload sizes and the specified configurations. They help to verify that the packet generation logic adheres to the necessary standards for padding, size limits, and overall data integrity.