Overview

This folder contains test cases and a test program used to demonstrate the validity of Gaussian-approximated statistical timing analysis. Three gate models are provided in the .cir files, and the corresponding .txt files contain delay data extracted from SPICE simulations.

A simple test case, as described in the report, is included and can be executed by running main.py. The same critical path is analyzed using both full Monte Carlo simulation and the Gaussian approximation method. The resulting delay distributions are displayed individually and overlaid for comparison.

The remaining 2 python files are used as debug utilities.

File: main.py

Purpose

This script performs statistical timing verification using three techniques: full Monte Carlo simulation, analytical Gaussian modeling, and hybrid sampling. It evaluates critical path delays in digital circuits and estimates the maximum safe operating frequency under uncertainty (e.g., process variation).

Key Components

1. TimingGraph

- Simulates arbitrary gate delays (tuples, sample lists, or sampling functions).
- **Methods:**
    - add_path(src, dst, gate_delays): Adds a directed edge with delay info.
    - sample_delay(gate): Samples delay based on its type.
    - simulate(clock_period, num_trials): Monte Carlo simulation to evaluate timing violations.
    - find_max_frequency(...): Binary search to find the highest safe frequency.
    - compute_edge_slacks(clock_period): Calculates slack for each edge.
    - visualize_delay_histograms(...): Plots histograms of edge delays.

2. MonteCarloTimingGraph

- Performs full path-level Monte Carlo timing analysis using sampled delays.
- **Methods:**
    - add_path(start_node, end_node, delays): Adds an edge with sample delays.
    - load_from_file(filename): Loads graph edges and delays from a file.
    - find_critical_path(): Finds the longest path based on average delays.
    - calculate_total_delay(path, num_trials): Samples and sums delays over trials.
    - find_maximum_frequency(): Calculates frequency from 99.7th percentile delay.

3. GaussianTimingGraph

- Assumes all delays are Gaussian.
- **Methods:**
    - add_path(src, dst, gate_delays): Adds Gaussian delays to an edge.
    - compute_total_delays(): Computes mean and stddev of each edge.
    - visualize_pdf(delay_stats): Plots Gaussian PDF for delays.
    - find_max_frequency(...): Computes max frequency using analytical Gaussian statistics.

Utility Functions

- read_samples(filename): Reads tab-separated sample delays from file.
- compute_mean_std(samples): Returns mean and standard deviation of a list.
- plot_monte_carlo_pdf(mc_total_delays): Plots a histogram of sampled delays.
- plot_overlay(mc_total_delays, total_mean, total_std): Overlays histogram with Gaussian PDF.

---

gaussian_test()

- Constructs a small test circuit with XOR → AND → OR → OUT using Gaussian delays.

- Computes path delay statistics and visualizes the total Gaussian delay distribution.
- Calculates max safe clock frequency analytically using Gaussian properties.