**AUTOMATED STATIC TIMING ANALYSIS TOOL FOR DIGITAL ELECTRONIC CIRCUITS**

**INTRODUCTION:**

* Static timing analysis is a crucial step in digital design to ensure that signals propagate correctly through the circuit within specified timing constraints.
* This project aims to develop an automated tool using Python for performing Static Timing Analysis (STA) on digital electronic circuits.

**OBJECTIVE:**

* Develop a Python-based tool to read circuit descriptions in netlist format.
* Calculate setup and hold times for flip-flops and identify timing violations.
* Provide solutions to rectify identified timing violations.
* Ensure the circuit design is DFT (Design for Testability) friendly.

**TARGET AUDIENCE:**

This project is designed primarily for students and researchers with a background in electronics and Very-Large-Scale Integration (VLSI) design. The key target audience includes:

* **Electronics Students:** Undergraduate and graduate students studying electronics, focusing on digital design, circuit analysis, and timing analysis.
* **VLSI Design Students:** Students specializing in VLSI design, who need to understand the intricacies of static timing analysis (STA) and timing violations in digital circuits.

**SOFTWARE USED:**

* **MULTISIM SOFTWARE:** Used to generate the circuit where STA is to be performed
* **PSPICE SOFTWARE:** Used to extract the netlist file of the circuit that is generated in the Multisim software
* **SPYDER:** IDE For Python development.

**TEST CASES:**

9 self-generated test circuits were executed for the following

* Combinational logic between the launch and capture flops
* Adding a MUX to the launch flop to check if the circuit is DFT (Design for testability) friendly
* Simulated for edge triggered component (latch) in substitution for level triggered component (flip flop)

**HOW TO USE THE TOOL**

1. **Create and Simulate the Circuit:**
   * Using Multisim tool, the schematic diagram that is targeted to be analysed is constructed. All the necessary components, such as logic gates, flip-flops, and connections, are to be included.
2. **Generate the Netlist:**
   * After successful simulation, netlist is generated from the Multisim file. This netlist file will automatically open in the PSPICE tool from Cadence for further analysis.
3. **Simulation Run** 
   * Simulation takes the following input
     1. File path that contains netlist file
     2. Setup time
     3. Hold time
   * Simulation generates the report for
     1. Timing violation
     2. Proposal to correct the violation

**NOTE:** The default clock to Q delay and clock frequency in the Multisim tool are 2 ns. In the Python code, there is a provision to adjust the clock frequency and clock to Q delay as per the research needs. This section can be commented to use the default value [2 ns].

**TAKING IT FORWARD**

* Addition of more combinational and sequential components to the circuit
* More complicated and large-scale circuits to be evaluated
* Providing a Graphical interface for this tool

**NOTE:** The step to use this tool and be found in README. In the repository