Cadence Genus Synthesis Guide

Introduction

This document provides a comprehensive guide to using the Cadence Genus synthesis tool for digital design implementation. It covers key aspects of the synthesis process, including input requirements, design elaboration, optimization strategies, and report generation. This guide is structured to help users efficiently integrate and customize the synthesis script for their specific projects.

1. Overview of Genus Synthesis

Cadence Genus is an advanced logic synthesis tool that converts RTL descriptions into gate-level representations optimized for performance, power, and area. The synthesis process involves several stages, including reading input files, elaborating the design, mapping logic to standard cells, and optimizing for timing and power.

2. Setting Up the Environment

Before executing the synthesis flow, the script sets up essential parameters:

- **Design Name**: The specific module or circuit being synthesized.
- **Synthesis Effort Levels**: Defines the trade-off between runtime and optimization quality.
- **Output Directories**: Organizes synthesized netlists and reports for traceability.
- **File Paths**: Specifies the location of required design files, including technology libraries and constraints.

Harsh A Patil
For any queries, feel free to contact me at harshpatilhp28@gmail.com.

3. Input File Requirements

The synthesis process relies on multiple input files:

- **Library Files (.lib)**: Contains technology-specific standard cell definitions.
- **LEF Files (.lef)**: Provides layout information for physical design.
- **RTL Files (.v)**: Represents the circuit in Verilog hardware description language.
- **SDC Files (.sdc)**: Defines design constraints such as clocking and timing requirements.

The script includes validation checks to ensure that all required files are available before proceeding.

4. Design Elaboration

Elaboration is the process of transforming the high-level RTL description into a detailed intermediate representation. This step resolves design hierarchy, checks for errors, and prepares the circuit for synthesis optimizations.

5. Synthesis Flow

The synthesis process consists of three key stages:

a) Logic Synthesis

Performs high-level optimization on the RTL to remove redundancies and generate an initial gate-level netlist.

b) Technology Mapping

Maps the synthesized logic onto available standard cells based on the specified technology library.

c) Optimization

Refines the mapped design to meet area, power, and timing constraints. This step applies transformations such as logic restructuring, buffer insertion, and gate sizing. Each stage generates an intermediate netlist for verification and debugging.

6. Report Generation

To evaluate the synthesized design, the script produces detailed reports:

- **Area Report**: Provides information about the physical footprint of the synthesized circuit.
- **Timing Report**: Analyzes timing paths and highlights violations.
- **Power Report**: Estimates dynamic and static power consumption.
- **Gate Count Report**: Lists the number of logic gates used, helping assess design complexity.

These reports enable designers to fine-tune the synthesis parameters for improved results.

7. Final Outputs and Verification

Once synthesis is complete, the final netlist is stored along with reports for further analysis. The script provides a summary message indicating successful execution and the location of saved files.

Best Practices for Effective Synthesis

To maximize synthesis efficiency and achieve optimal results, consider the following best practices:

- Ensure proper constraint definition in SDC files for accurate timing analysis.
- Use appropriate synthesis effort levels to balance performance and runtime.
- Regularly review reports to identify areas for improvement in power, area, and timing.

Harsh A Patil
For any queries, feel free to contact me at harshpatilhp28@gmail.com.

- Validate the synthesized netlist through functional and gate-level simulations before proceeding to physical design.
- Modify the synthesis script accordingly to reflect changes in design constraints, optimization strategies, or file paths for improved results.

By following these guidelines, users can enhance their digital design workflow and achieve optimized implementations with Cadence Genus.

Harsh A Patil
For any queries, feel free to contact me at harshpatilhp28@gmail.com.