VHDL Code Generator Project Report

Introduction

This project is a graphical user interface (GUI) application developed using Python's Tkinter library. It is designed to generate VHDL (VHSIC Hardware Description Language) code for various digital components commonly used in digital design. The application supports combinational circuits such as Multiplexers (MUX), Demultiplexers (DEMUX), Decoders, and Encoders, as well as sequential circuits like Parallel-In Serial-Out (PISO) and Serial-In Parallel-Out (SIPO) shift registers.

Objectives

- To provide an easy-to-use GUI for generating VHDL code without manual coding.
- To support multiple digital components with configurable parameters.
- To include optional features like ENABLE signals or control signals (Clock, Reset, Mode).
- To validate user inputs and guide users through feature selection.
- To dynamically generate VHDL code tailored to user specifications.

Features and Functionality

1. Component Detection and Code Generation

The application automatically detects the type of digital component based on user inputs:

- **MUX:** Detected when multiple inputs map to a single output.
- **DEMUX:** Detected when a single input maps to multiple outputs.
- **Decoder:** Recognized when the number of outputs equals 2^number_of_inputs.
- **Encoder:** Recognized when the number of inputs equals 2^number_of_outputs.

Depending on the detected component, the program prompts the user about optional signals like ENABLE and then generates the corresponding VHDL code.

2. Optional Features Dialog

For some components, the application shows a confirmation dialog to include or exclude features such as ENABLE, MODE, CLK, and RESET signals, enabling the user to customize the generated code.

3. VHDL Code Generation

The application produces well-structured VHDL code with appropriate entity and architecture declarations. It uses parameterized inputs, output sizes, and includes processes and case statements for combinational logic. For sequential components (PISO/SIPO), it supports clocked processes with reset and mode control.

4. Shift Registers (PISO and SIPO)

The second tab focuses on shift registers. Based on the "MODE" checkbox:

- If MODE is selected, the application generates a PISO shift register.
- Otherwise, it generates a SIPO shift register.

Both require clock and reset signals to be selected for code generation. The code includes the necessary processes for shifting and loading data.

5. User Interface Design

- The GUI uses Tkinter Notebook tabs to separate combinational and sequential components.
- Input fields allow specifying the number of inputs, outputs, select signals, and bit widths.
- Checkboxes let users select optional control signals.
- A text box displays the generated VHDL code, with syntax formatting preserved.

6. Error Handling

The program includes input validation and error dialogs to ensure only valid configurations are accepted, such as correct numbers of select signals and positive integers for inputs and bit widths.

Technical Details

- **Language:** Python 3.x
- **GUI Framework:** Tkinter
- **Mathematical Computation:** `math.log2` to calculate required select signals
- **Modularity:** Functions are modularized for generating different components' VHDL code.
- **Dialogs:** Custom yes/no dialogs for user confirmation.
- **Output:** VHDL code is displayed in a read-only text widget.

Conclusion

This project successfully automates the generation of VHDL code for various common digital components, improving productivity and reducing manual coding errors. It offers a user-friendly interface and flexible options for different design needs, making it a useful tool for students and engineers working with VHDL.