Simple Language Parser Documentation

Project Overview

This project creates a basic parser for a custom hardware description language using flex (lex) and bison (yacc). The parser checks the syntax and meaning of the code according to the rules of the language.

File Structure

- lexer.l: Contains the rules for breaking down the text into tokens (words).
- parser.y: Contains grammar rules and actions to check the meaning of the code.
- testingScript.sh: A script that runs test cases to check if the parser works correctly.
- Makefile:

Language Features

1. Case-insensitive: The language does not distinguish between uppercase and lowercase letters.
2. Single entity/architecture: Each file can only contain one entity and one architecture.
3. Identifiers: Names (like entity, signal, or architecture names) must start with a letter or an underscore (_) and can include letters, numbers, and underscores.
4. Signal declarations: Signals may or may not be declared.
5. Assignments: Signals may or may not be assigned values.
6. Entity consistency: The architecture must refer to the correct entity name.
7. Signal rules: Only previously declared signals can be used, and signal assignments must be of the same type.

Implementation Details

Lexical Analysis (lexer.l)

The lexer identifies:

- Keywords: ENTITY, ARCHITECTURE, SIGNAL, IS, OF, BEGIN, END
- Operators: The assignment operator <=
- Identifiers: Valid names (and invalid ones)
- Whitespace and newlines: Ignored by the parser
- Special characters: Like : = ;

Syntax and Semantic Analysis (parser.y)

The parser includes:

- Symbol Table: Keeps track of signals and their types.
- Error Reporting: Gives clear error messages for mistakes in the code.
- Grammar Rules: Defines how the language should be structured, such as how entities and architectures should be written.

Key Checks:

- Make sure the entity name is the same in both the declaration and architecture.
- Ensure signals are declared before being used.
- Check that signals being assigned to each other are of the same type.
- Error Handling

The parser reports errors for:

1. Invalid names
2. Duplicate signals
3. Using undeclared signals
4. Type mismatches in assignments
5. Entity name mismatches