LogicXplorer

Electronic Design Automation Tool

For Verilog/SystemVerilog Analysis and Testbench Generation

Complete User Manual

Version 1.0

December 29, 2025

Developer: Kunal Saraswat

kunalsaraswat30@gmail.com

Copyright Notice

Document Version: 1.0
Release Date: December 29, 2025

Contents

List of Tables

CHAPTER 1

Introduction to LogicXplorer

1.1 Overview

LogicXplorer is a professional Electronic Design Automation (EDA) tool designed specifically for Verilog and SystemVerilog analysis. It provides comprehensive capabilities for RTL analysis, testbench generation, hierarchy visualization, and gate-level netlist examination. Built with Python and Tkinter, it offers an intuitive graphical interface for hardware design engineers.

1.2 Key Features

LogicXplorer incorporates several advanced features:

- **RTL Analysis:** Automated extraction and analysis of Verilog/SystemVerilog modules
- **Hierarchy Visualization:** Interactive tree views and schematic diagrams
- **Testbench Generation:** Automatic UVM, AXI, and APB testbench creation
- **Gate-Level Analysis:** Examination and explanation of synthesized netlists
- **Protocol Detection:** Automatic detection of AXI, APB, and other bus protocols
- **Theme Support:** Multiple visual themes for user preference
- **Large File Support:** Optimized performance for multi-megabyte design files
- **Project Management:** Multi-file project handling with intelligent merging

1.3 Target Audience

This tool is designed for:

- Hardware Design Engineers
- Verification Engineers
- FPGA/ASIC Designers
- Students learning hardware description languages
- Research professionals in electronic design
- CAD/EDA tool developers

Information

System Requirements:

- Python 3.7 or higher
- 4GB RAM minimum (8GB recommended)
- 500MB disk space
- Windows/Linux/macOS with Tkinter support

CHAPTER 2

Installation and Setup

2.1 Prerequisites

Before installing LogicXplorer, ensure the following prerequisites are met:

Step 1: Python Installation

1. Download Python 3.7 or higher from `python.org`
2. During installation, select "Add Python to PATH"
3. Verify installation: `python --version`

Step 2: Required Libraries

1. Install required packages:

```
pip install tkinter pip install pygments
```

2. For additional features (optional):

```
pip install matplotlib pip install graphviz
```

2.2 Installation Procedure

Follow these steps to install LogicXplorer:

1. **Download the Tool:** Obtain the latest version from the official repository
2. **Extract Files:** Unzip the package to your preferred directory
3. **Verify Structure:** Ensure the following files are present:
   - `TEST_1.py` (Main application)
   - `requirements.txt` (Dependencies)
   - `README.md` (Documentation)
4. **Install Dependencies:**

```
pip install -r requirements.txt
```

5. **Launch Application:**

```
python TEST_1.py
```

2.3 First-Time Configuration

Upon first launch, configure the following:

Table 2.1: Initial Configuration Settings

Setting	Description	Default Value
Theme	Visual theme for the interface	Cyborg
Default Directory	Initial directory for file operations	User's Home
Font Size	Code editor font size	11
Auto-save Interval	Minutes between auto-saves	5
Max Recent Files	Number of files in recent list	8
Syntax Highlighting	Enable/disable syntax coloring	Enabled
Thread Workers	Number of parallel analysis threads	4

2.4 Profile Activation

To access all features, activate your license:

1. Click the **Profile** button in the toolbar
2. Enter your email address
3. Input the 12-digit license key
4. Click **Activate**
5. The status will confirm successful activation

Warning

License Requirements:

- A valid license key is required for commercial use
- Educational licenses are available for academic institutions
- Trial licenses provide limited functionality for 30 days

CHAPTER 3

User Interface Guide

3.1 Main Window Layout

The LogicXplorer interface is organized into several key areas:

3.1.1 Menu Bar

The menu bar provides access to all major functions:

Table 3.1: Menu Bar Functions

Menu	Functions
File	New file, Open file, Open project, Recent files, Exit
Edit	Undo, Redo, Find text, Replace
View	Zoom controls, Theme selection, Toolbar toggle
Analyze	Testbench generation, Design reports, Header preview
Visualization	Hierarchy views, Schematic diagrams, Block diagrams
Netlist	Gate-level netlist analysis and visualization
Help	Documentation, About information

3.1.2 Toolbar

The toolbar provides quick access to frequently used functions:

1. **Load File:** Open individual Verilog/SystemVerilog files
2. **Load Project:** Open entire project directories
3. **Explain Code:** Generate code analysis report
4. **Copy Explanation:** Copy analysis to clipboard
5. **Clear All:** Reset workspace
6. **Cancel:** Stop ongoing operations
7. **Profile:** License activation and management
8. **Theme Selector:** Change visual theme

3.1.3 Notebook Tabs

The main workspace uses a tabbed interface:

Code Tab

- File selector dropdown for multi-file projects
- Line number display
- Syntax-highlighted code editor
- Support for large files with optimized rendering

Explanation Tab

- Detailed analysis output
- Formatted reports with sections
- Copy functionality for sharing
- Persistent across file switches

Additional Tabs

- **Testbench:** Generated testbench code
- **Schematic:** Interactive circuit diagrams
- **Hierarchy:** Module relationship trees

3.2 Theme Selection

LogicXplorer offers multiple visual themes:

Table 3.2: Available Themes

Theme	Primary Color	Description
Cyborg	Blue	Dark theme with blue accents (Default)
Darkly	Dark Gray	Professional dark theme
Solar	Dark Blue	Solarized dark variant
Superhero	Blue-Green	High contrast theme
United	Orange	Light theme with orange accents

To change themes:

1. Select **View** from menu bar
2. Use the theme selector in toolbar
3. Changes apply immediately

3.3 Keyboard Shortcuts

Efficient navigation using keyboard shortcuts:

Table 3.3: Keyboard Shortcuts

Shortcut	Function
Ctrl+N	New file
Ctrl+O	Open file
Ctrl+S	Save file
Ctrl+Shift+S	Save As
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+F	Find text
Ctrl+H	Replace text
Ctrl++	Zoom in
Ctrl+-	Zoom out
Ctrl+0	Reset zoom
F1	Help documentation
F5	Refresh analysis
Alt+G	Generate testbench

CHAPTER 4

File and Project Management

4.1 Working with Individual Files

LogicXplorer supports various file operations:

4.1.1 Opening Files

1. Click **Load File** in toolbar or use Ctrl+O
2. Navigate to Verilog/SystemVerilog file
3. Supported extensions: .v, .sv, .vh, .svh
4. Large files load with progress indication

4.1.2 File Encoding

The tool automatically handles:

- UTF-8 encoding (primary)
- ASCII fallback
- Unicode character support
- Line ending normalization

4.1.3 Recent Files

- Track up to 8 recently opened files
- Access via File → Open Recent
- Click to reload file
- List persists between sessions

4.2 Project Management

For complex designs with multiple files:

4.2.1 Creating Projects

1. Click **Load Project** in toolbar
2. Select directory containing design files
3. Tool scans for Verilog/SystemVerilog files
4. Files are indexed and loaded

4.2.2 Project Structure

The tool expects this structure:

```
project_folder/rtl/module1.vmodule2.svmodule3.vtb/testbench.svconstraints/timing.sdc
```

4.2.3 File Switching

Within a project:

- Use dropdown in Code tab
- Instant switching between files
- Each file maintains its view state
- Cross-file references are tracked

4.3 Code Editor Features

The integrated editor provides:

4.3.1 Syntax Highlighting

- Keywords: module, always, assign, etc.
- Data types: wire, reg, logic, etc.
- Comments: single-line and multi-line
- Strings and numbers
- SystemVerilog extensions

4.3.2 Line Numbering

- Continuous line numbering
- Click to select lines
- Right-click for line operations
- Search by line number

4.3.3 Text Operations

- Find and replace
- Case-sensitive search
- Regular expressions

- Go to line
- Text selection modes

4.4 Performance Considerations

For optimal performance with large files:

Table 4.1: File Size Recommendations

File Size	Performance	Recommendations
¡ 1MB	Excellent	Full feature access
1-10MB	Good	Moderate analysis depth
10-50MB	Fair	Limit analysis scope
¿ 50MB	Basic	View-only recommended

Information

Performance Tips:
- Close unused files in project
- Use "Quick Analysis" for large files
- Disable auto-syntax highlighting for ¿10MB files
- Increase memory allocation if available

CHAPTER 5

Code Analysis Features

5.1 Automatic Module Extraction

LogicXplorer automatically detects and extracts modules:

5.1.1 Detection Algorithm

The tool uses regex patterns to identify:

- Module declarations
- Port lists and parameters
- Nested modules
- Testbench modules

5.1.2 Module Information

For each detected module, the tool extracts:

1. Module name
2. Port directions (input/output/inout)
3. Port ranges and data types
4. Parameter values
5. Internal instances
6. Always blocks and assignments

5.2 Hierarchy Analysis

The tool builds design hierarchy:

5.2.1 Hierarchy Detection

```
Example hierarchy detection module top(); moduleA inst1(...); moduleB inst2(...);
moduleC inst3(...); endmodule
```

5.2.2 Top Module Identification

- Analyzes instantiation patterns
- Identifies top-level modules
- Filters testbench modules
- Determines design root

5.3 Protocol Detection

Automatic bus protocol identification:

5.3.1 AXI Protocol

Table 5.1: AXI Signal Detection

Signal Type	Example Signals	Detection Pattern
Clock	ACLK, aclk	clk, clock, aclk
Reset	ARESETn, reset_n	rst, reset, aresetn
Address	AWADDR, ARADDR	addr, awaddr, araddr
Data	WDATA, RDATA	data, wdata, rdata
Control	AWVALID, WREADY	valid, ready

5.3.2 APB Protocol

- PCLK detection
- PRESETn detection
- PADDR, PWDATA, PRDATA
- PSEL, PENABLE, PREADY

5.4 Design Metrics

Comprehensive design statistics:

5.4.1 Code Metrics

Table 5.2: Design Metric Calculations

Metric	Calculation Method
Lines of Code	Total non-empty lines
Module Count	Unique module declarations
Instance Count	Total instantiations
Always Blocks	Always statements
Assign Statements	Continuous assignments
Port Count	Inputs + Outputs + Inouts
Parameter Count	Module parameters
Hierarchy Depth	Maximum nesting level

5.5 Analysis Reports

Generated reports include:

5.5.1 Report Structure

I. **Header:** File/project name and timestamp

II. **Module Summary:** List of detected modules

III. **Top Module:** Design hierarchy root

IV. **Module Details:** Per-module statistics

V. **Protocol Detection:** Bus interfaces found

VI. **Design Metrics:** Quantitative analysis

VII. **Recommendations:** Design suggestions

5.5.2 Customization Options

- Analysis depth (Quick/Detailed)
- Include/exclude testbenches
- Focus on specific modules
- Export format selection

Information

Analysis Depth Settings:
- **Quick:** Surface-level analysis (fast)
- **Detailed:** Full analysis (comprehensive)
- **Custom:** User-selected analysis components

CHAPTER 6

Visualization Features

6.1 Hierarchy Views

Interactive hierarchy exploration:

6.1.1 Tree View

- Expandable/collapsible tree structure
- Color-coded module types
- Instance naming display
- Click-to-navigate functionality

6.1.2 Graph Visualization

- Force-directed graph layout
- Module relationship visualization
- Zoom and pan capabilities
- Export as image/PDF

6.2 Schematic Generation

Professional schematic diagrams:

6.2.1 Block Diagrams

Table 6.1: Schematic Component Types

Component	**Visual Style**	**Purpose**
Ports	Colored rectangles	Interface signals
Modules	Rounded boxes	Design modules
Instances	Rectangular boxes	Module instances
Wires	Colored lines	Signal connections
Buses	Thick lines	Multi-bit connections
Constants	Diamond shapes	Fixed values

6.2.2 Interactive Features

- Click components for details
- Highlight signal paths
- Toggle visibility of nets
- Search within schematic

6.3 Module Graph View

Detailed module-level visualization:

6.3.1 Input/Output Display

- Left panel: Input ports
- Center panel: Internal signals
- Right panel: Output ports
- Color coding by signal type

6.3.2 Zoom and Navigation

1. Mouse wheel zoom
2. Pan with drag
3. Fit-to-view button
4. Zoom percentage display

6.4 Theme-Aware Visualization

Consistent with application theme:

6.4.1 Color Schemes

```
Theme color mapping for visualization visualization_colors = 'cyborg' : 'background' : 0b0b0b','module_fill' : 2A9F
```

6.4.2 Customization Options

- Node size adjustment
- Font size scaling
- Line width settings
- Color scheme selection

6.5 Export Capabilities

Visualization export options:

6.5.1 Supported Formats

Table 6.2: Export Format Support

Format	Resolution	Use Case
PNG	Up to 4K	Documentation
PDF	Vector	Printing
SVG	Vector	Editing
JPEG	Adjustable	Web use
BMP	Lossless	Legacy systems

6.5.2 Export Settings

- Page size (A4, Letter, etc.)
- Orientation (Portrait/Landscape)
- Margin settings
- Background inclusion
- Grid display options

Warning

Large Schematic Considerations:
- Schematics with ¿1000 nodes may render slowly
- Consider exporting in sections
- Use "Simplify" option for complex designs
- Increase memory allocation for large exports

CHAPTER 7

Testbench Generation

7.1 UVM Testbench Generation

Professional UVM testbench creation:

7.1.1 Template Structure

```
// UVM Testbench Structure `timescale 1ns/1ps
interface tb_if; //Interfacesignalsendinterface
package tb_pkg; importuvm_pkg :: *;
class transaction extends uvm_sequence_item; //Transactionfieldsendclass
class driver extends uvm_driver(transaction); //Driverimplementationendclass
// Additional UVM components endpackage
module tb_top; //Testbenchtop − levelendmodule
```

7.1.2 Generated Components

Table 7.1: UVM Component Generation

Component	**Generated Features**
Interface	Signal declarations with modports
Transaction	Randomized fields with constraints
Driver	Sequence item processing
Monitor	Signal sampling and analysis
Sequencer	Sequence execution control
Agent	Component integration
Environment	Testbench orchestration
Test	Test case implementation

7.2 AXI Testbench Generation

AXI4/AXI4-Lite testbench creation:

7.2.1 AXI Signal Mapping

```
// AXI Signal detection and mapping axi_mapping = 'clock' : 'ACLK','reset' : 'ARESETn','awaddr' : 'AWADDR','
```

7.2.2 AXI Test Sequences

1. Single write transaction
2. Single read transaction
3. Sequential writes
4. Sequential reads
5. Random burst patterns
6. Error condition testing
7. Back-pressure scenarios

7.3 APB Testbench Generation

APB protocol testbench creation:

7.3.1 APB Components

- APB interface with all signals
- APB master driver
- APB slave model
- APB monitor
- APB scoreboard

7.3.2 APB Test Coverage

Table 7.2: APB Test Coverage

Test Type	Coverage	Description
Basic Transfer	100%	Single read/write
Back-to-Back	95%	Continuous transfers
Wait States	90%	PREADY delays
Error Response	85%	PSLVERR generation
Address Boundaries	100%	Edge cases
Random Sequence	80%	Random operations

7.4 Customization Options

Testbench generation settings:

7.4.1 Generation Parameters

Table 7.3: Testbench Generation Parameters

Parameter	Options	Default
Testbench Type	UVM, AXI, APB, Basic	UVM
Clock Period	1-100ns	10ns
Reset Duration	10-1000ns	100ns
Test Length	Short, Medium, Long	Medium
Assertion Level	None, Basic, Full	Basic
Coverage Collection	Disabled, Enabled	Disabled
Waveform Dump	None, VCD, FSDB	VCD

7.4.2 Advanced Features

- **Parameter Passing:** Correct DUT parameter instantiation
- **Interface Generation:** Protocol-specific interfaces
- **Scoreboard Integration:** Automatic response checking
- **Clock/Reset Generation:** Configurable timing
- **Stimulus Patterns:** Pre-defined test sequences
- **Error Injection:** Controlled error generation

7.5 Output Management

Generated testbench handling:

7.5.1 Saving Options

1. Save as SystemVerilog (.sv)
2. Save as Verilog (.v)
3. Save with project context
4. Save individual components

7.5.2 Integration Features

- Direct insertion into project
- Component file separation
- Makefile generation
- Simulation script creation

Information

Best Practices:
- Generate testbench early in design cycle
- Review auto-generated code before use
- Customize for specific verification needs
- Maintain version control of testbenches

CHAPTER 8

Gate-Level Netlist Analysis

8.1 Netlist Loading and Parsing

Processing gate-level designs:

8.1.1 Supported Formats

Table 8.1: Netlist Format Support

Format	Extensions	Features
Verilog Netlist	.v, .sv	Full support
EDIF	.edf, .edif	Basic support
SPICE	.sp, .cir	Limited support
Cadence Genus	.v (Genus)	Enhanced parsing
Synopsys DC	.v (DC)	Standard parsing

8.1.2 Parsing Engine

- Regex-based cell extraction
- Hierarchical netlist support
- Parameter and attribute handling
- Comment and directive filtering

8.2 Cell Library Analysis

Standard cell examination:

8.2.1 Cell Classification

```
Cell type classification cell_categories = 'FLIP_FLOPS' : ['DFF','SDFF','DFFR'],'LATCHES' : ['LATCH',
```

8.2.2 Cell Statistics

Table 8.2: Cell Statistics Analysis

Metric	Calculation	Purpose
Total Cells	Count all instances	Design size
Unique Cell Types	Count distinct types	Library usage
Gate Count	Weighted sum	Complexity measure
Cell Distribution	Percentage per type	Design composition
Max Fanout	Maximum load per net	Timing analysis
Average Fanout	Mean load per net	Quality metric

8.3 Connectivity Analysis

Net and connection examination:

8.3.1 Net Extraction

- Signal net identification
- Power/ground net filtering
- Clock network analysis
- Reset network analysis

8.3.2 Connectivity Metrics

1. Driver-load relationships
2. Fanout calculations
3. Critical net identification
4. Floating net detection
5. Multi-driven net detection

8.4 Critical Path Analysis

Timing path examination:

8.4.1 Path Extraction

- Startpoint identification (registers/inputs)
- Endpoint identification (registers/outputs)
- Combinational path tracing
- Clock path analysis

8.4.2 Path Metrics

Table 8.3: Critical Path Metrics

Metric	Unit	Description
Path Length	Cells	Number of cells in path
Estimated Delay	ns	Rough timing estimate
Cell Types	List	Cells in critical path
Slack	ns	Timing margin (if known)
Worst Fanout	Count	Maximum loading in path

8.5 Visualization of Gate-Level

Schematic views for netlists:

8.5.1 Netlist Schematic

- Cell placement algorithms
- Net routing visualization
- Color coding by cell type
- Interactive component selection

8.5.2 Zoom and Detail Levels

1. Full-chip view
2. Module-level view
3. Cell-level view
4. Net-level view

8.6 Reporting and Export

Analysis result presentation:

8.6.1 Report Generation

- Summary statistics
- Cell library analysis
- Connectivity report
- Critical path report
- Recommendations

8.6.2 Export Formats

- HTML report
- CSV data export
- JSON structured data
- PDF summary

Warning

Gate-Level Analysis Limitations:

- No actual timing information without SDF
- Power analysis requires additional data
- Physical layout effects not considered
- Complex hierarchical designs may have parsing issues

Index

LogicXplorer

EDA Tool for Verilog/SystemVerilog Analysis

End of Manual

Document ID: LX-MANUAL-1.0
Revision: A
Printed: December 29, 2025