AMBA Bus Matrix Configuration Tool

User Guide and Reference Manual

Version 1.0.0

July 2025

Table of Contents

1. Introduction

The AMBA Bus Matrix Configuration Tool is a comprehensive solution for designing and implementing ARM AMBA-based System-on-Chip (SoC) interconnects. This tool provides both a graphical user interface for visual design and a powerful backend for generating synthesizable RTL and verification environments.

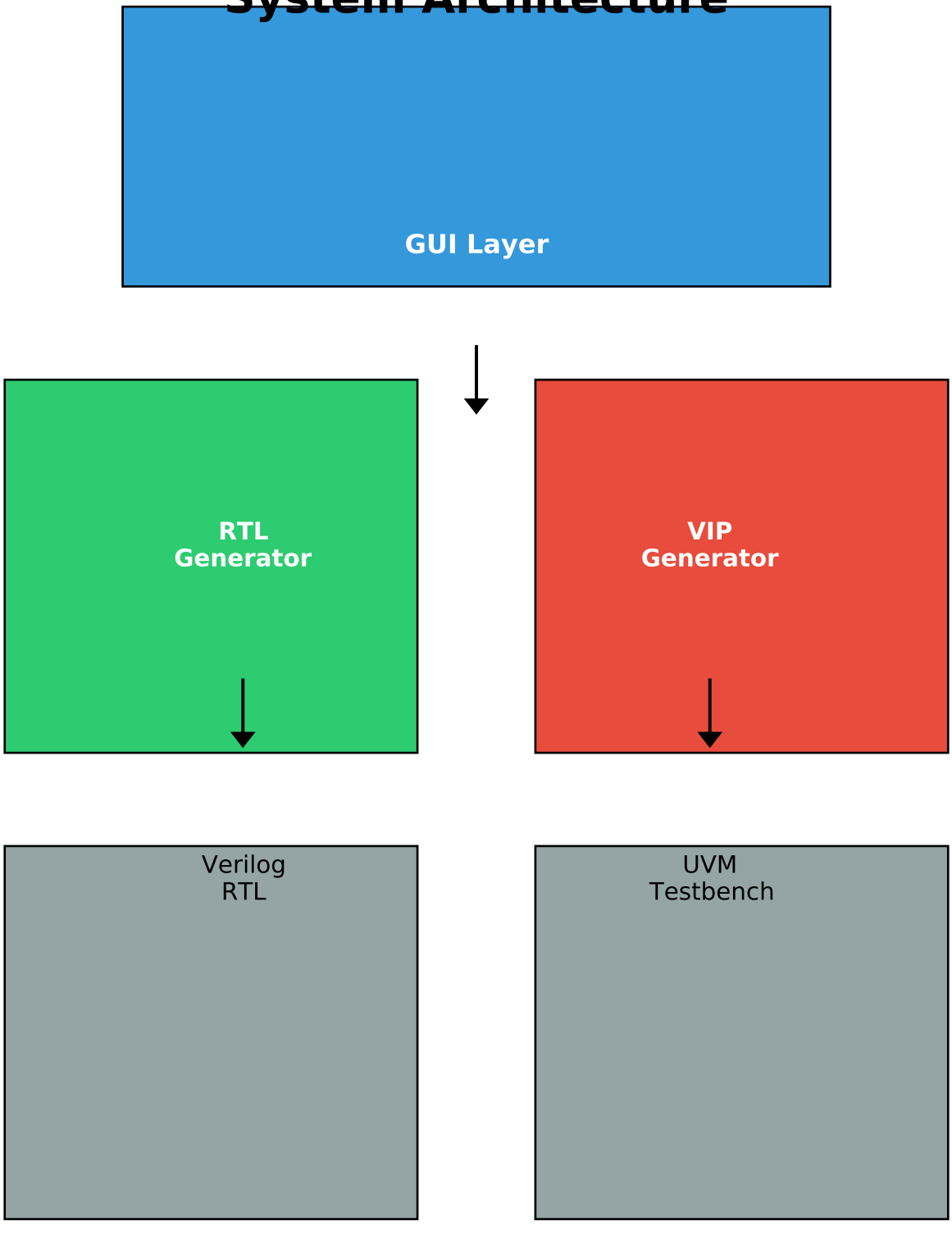
1.1 Key Features

- Visual bus matrix design with drag-and-drop interface
- Support for AXI4, AXI3, AHB, and APB protocols
- Automatic RTL generation with parameterizable configurations
- Complete UVM-based verification environment generation
- Built-in address overlap detection and validation
- Security and QoS configuration support

1.2 System Requirements

- Python 3.6 or higher
- Tkinter GUI library
- SystemVerilog simulator (VCS, Questa, or Xcelium)
- UVM 1.2 library

System Architecture



2. Getting Started

2.1 Installation

Clone the repository and install dependencies:

```
cd /your/project/directory
git clone <repository_url>
cd axi4_vip/gui
pip install -r requirements.txt
```

2.2 Quick Start

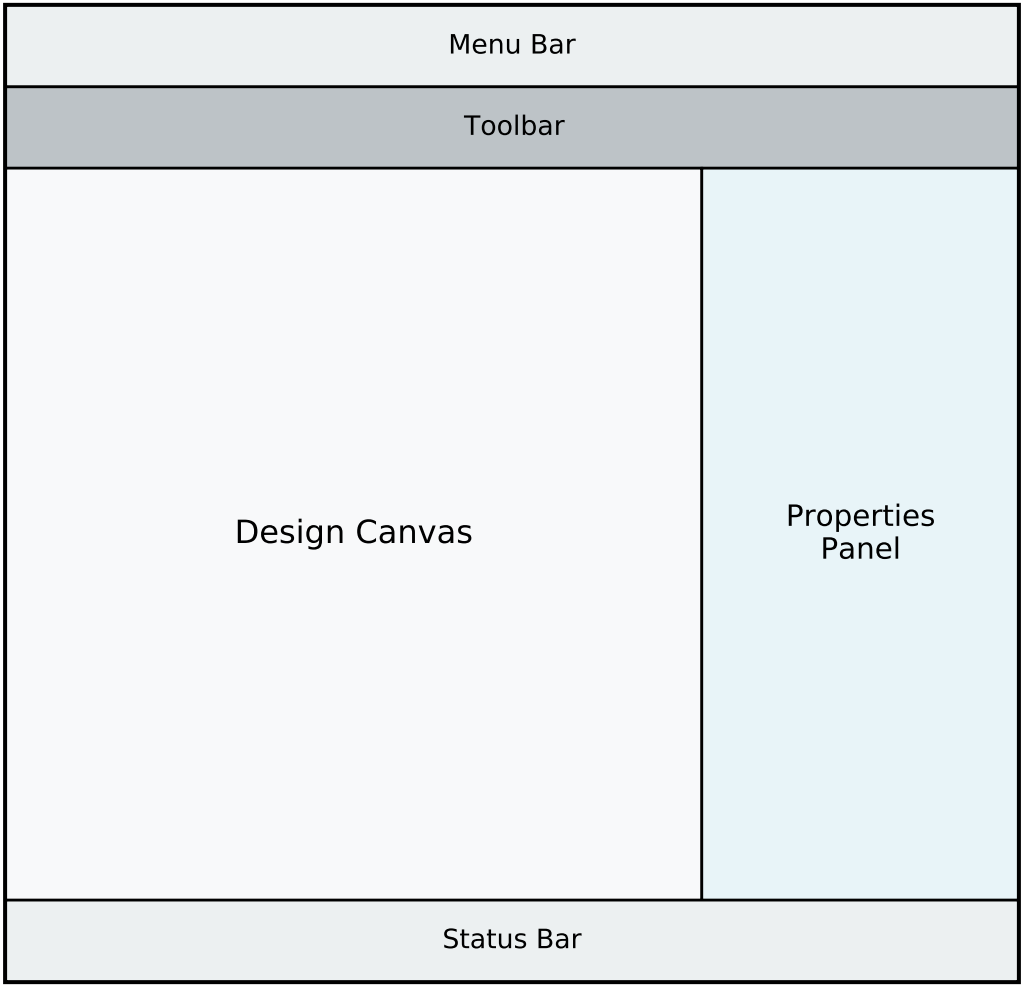
Launch the GUI and create your first design:

```
./launch_gui.sh
```

Follow these steps:

- Select the bus protocol (AXI4 is default)
- Click Add Master to add bus masters
- Click Add Slave to add bus slaves
- Draw connections between masters and slaves
- Configure addresses and parameters
- Click Generate RTL to create Verilog files

GUI Layout


Menu Bar
Toolbar
Design Canvas
Properties Panel
Status Bar

3. Creating Bus Designs

3.1 Adding Masters

Masters represent components that initiate transactions:

- CPU cores
- DMA engines
- GPU processors
- PCIe endpoints

3.2 Configuring Masters

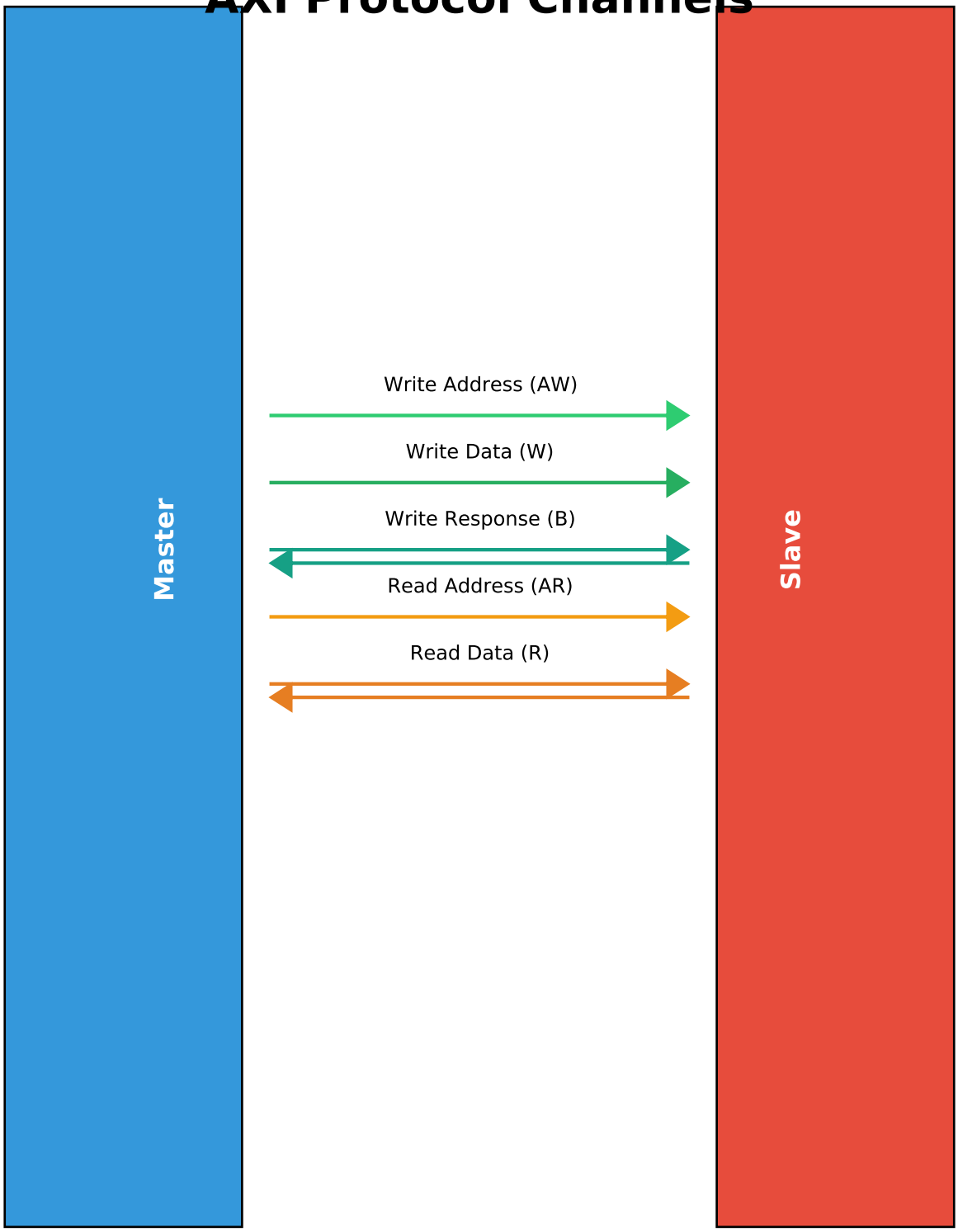
- Name: Descriptive identifier
- ID Width: Transaction ID bits
- Priority: Arbitration priority
- QoS Support: Enable quality of service
- Exclusive Support: Enable exclusive access

3.3 Adding Slaves

Slaves respond to transactions:

- Memory controllers (DDR, SRAM)

AXI Protocol Channels



4. RTL Generation

4.1 Generated Files

- axi4_interconnect_mNsM.v - Top-level module
- axi4_address_decoder.v - Address decoding
- axi4_arbiter.v - Arbitration logic
- axi4_router.v - Transaction routing
- tb_axi4_interconnect.v - Basic testbench

4.2 Module Parameters

```
module axi4_interconnect_m2s3 #(
  parameter DATA_WIDTH = 128,
  parameter ADDR_WIDTH = 40,
  parameter ID_WIDTH   = 4
)
```

5. Troubleshooting

5.1 Common Issues

Port Width Mismatch Warnings:

```
Lint-[PCWM-L] Port connection width mismatch
```

Solution: Regenerate RTL with latest version

GUI Launch Issues:

```
ImportError: No module named tkinter
```

Solution: Install tkinter package

5.2 Debug Mode

Enable debug output:

```
export AXI_VIP_DEBUG=1
./launch_gui.sh --debug
```