YosysHQ SVA AXI Properties - Crossbar Example

June 28, 2021

Contents

Part I
Introduction

1 Preface

1.1 Description

The following example shows how to use the YosysHQ SVA AXI Formal IP to verify Alex Forencich's AXI4 Crossbar. It is demonstrated that, with the current status of development of the IP, several errors are detected in that crossbar design.

1.2 Scope and limitations

- Not all AXI4-full behaviors are covered yet, there is still work in progress for the data transport checks as well. **This guide is an early demonstration example.**
- The AXI4-full subset of properties are still being developed at the moment, so there exist some holes that are not yet covered by the formal IP.
- This example shows the usage of the IP, a recipe of SBY and a debugging methodology. Feedback for improvement is very welcome.

Note: The YosysHQ AXI4 SVA Formal IP covers the AMBA ARM specification IHI version 0022E.

Part II
Architecture of the Test

2 Verification Plan

To verify this crossbar, a simple source to destination (1S:1D) topology for the crossbar is selected for the sake of simplicity. This is done by setting both S_COUNT and M_COUNT parameters of the axi_crossbar.v file as follows:

```
    // Number of AXI inputs (slave interfaces)
    parameter S_COUNT = 1,
    // Number of AXI outputs (master interfaces)
    parameter M_COUNT = 1,
```

Figure 1: Configuring the axi_crossbar.v module.

By using two YosysHQ AXI4 agents, one acting as source (that will provide constraints for source inputs and assertions for source outputs), and other acting as destination (that will provide constraints for destination inputs and assertions for destination outputs) the crossbar invariant (or interface) functions will be monitored for any kind of failures. The following section shows a block diagram of this configuration.

3 Block Diagram

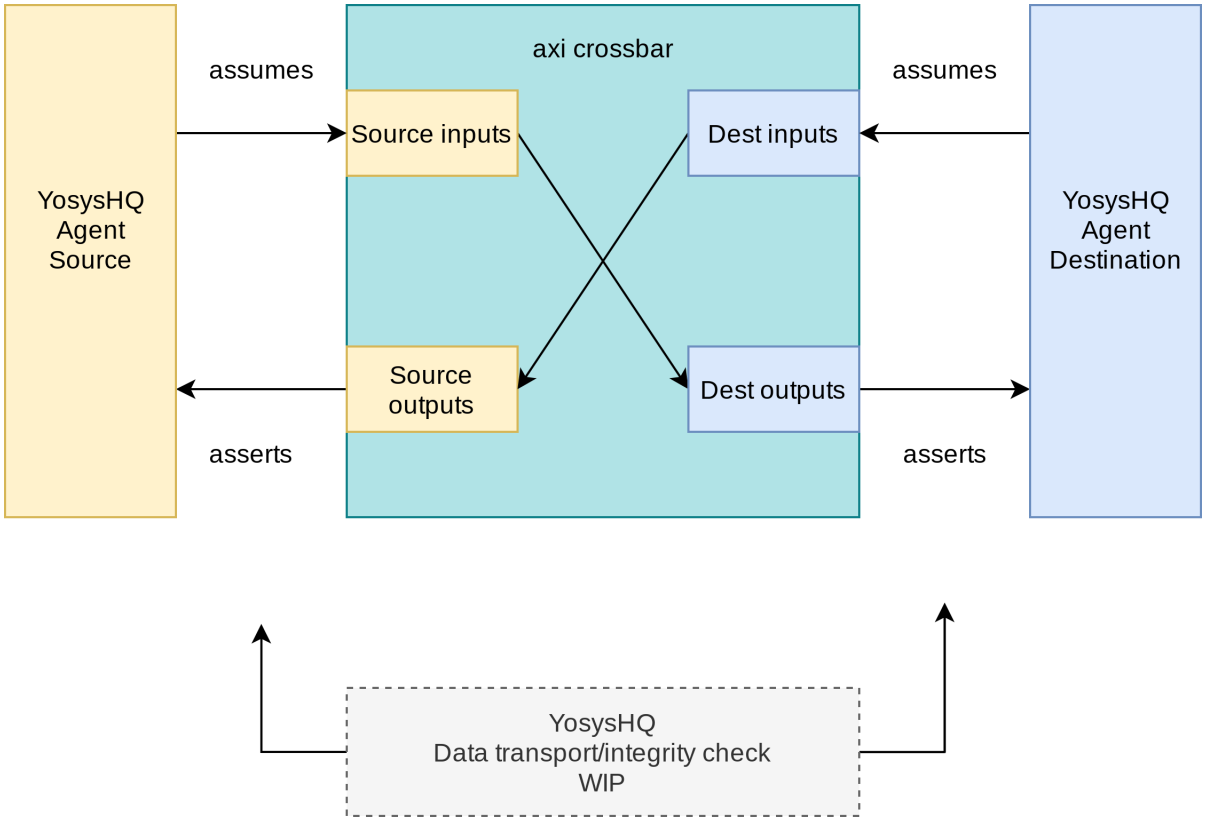

Figure 2: Block diagram of the AXI crossbar test.

Part III
TabbyCAD Setup for AXI4 Formal Crossbar Checks

4 The YosysHQ AXI4 Formal Configuration

The file **testbench.sv** instantiates and configures both **source** and **destination** agents as shown in Figure 2.

The source agent (to check for source behaviors) is configured as follows:

```
bind axi_crossbar amba_axi4_protocol_checker
  #('{ID_WIDTH:          4,
      ADDRESS_WIDTH:    32,
      DATA_WIDTH:       32,
      AWUSER_WIDTH:      1,
      WUSER_WIDTH:       1,
      BUSER_WIDTH:       1,
      ARUSER_WIDTH:      1,
      RUSER_WIDTH:       1,
      MAXWAIT:          20,
      AGENT_TYPE:       amba_axi4_protocol_checker_pkg::SOURCE,
      PROTOCOL_TYPE:    amba_axi4_protocol_checker_pkg::AXI4FULL,
      ENABLE_COVER:      1,
      ARM_RECOMMENDED:   0,
      CHECK_PARAMETERS:  1,
      OPTIONAL_WSTRB:    1,
      OPTIONAL_RESET:    0,
      EXCLUSIVE_ACCESS:  1})
source_chk (
            .ACLK(clk),
            .ARESETn(!rst),
```

Figure 3: Binding the YosysHQ AXI4 Formal IP to the crossbar as a source agent.

The destination agent (to check for destination behaviors) is configured as follows:

```
bind axi_crossbar amba_axi4_protocol_checker
  #('{ID_WIDTH:         4,
      ADDRESS_WIDTH:    32,
      DATA_WIDTH:       32,
      AWUSER_WIDTH:     1,
      WUSER_WIDTH:      1,
      BUSER_WIDTH:      1,
      ARUSER_WIDTH:     1,
      RUSER_WIDTH:      1,
      MAXWAIT:          20,
      AGENT_TYPE:       amba_axi4_protocol_checker_pkg::DESTINATION,
      PROTOCOL_TYPE:    amba_axi4_protocol_checker_pkg::AXI4FULL,
      ENABLE_COVER:     1,
      ARM_RECOMMENDED:  0,
      CHECK_PARAMETERS: 1,
      OPTIONAL_WSTRB:   1,
      OPTIONAL_RESET:   0,
      EXCLUSIVE_ACCESS: 1})
dest_check (
```

Figure 4: Binding the YosysHQ AXI4 Formal IP to the crossbar as a destination agent.

5 The SymbiYosys File

The main file is axi_crossbar.sby and the contents are described below.

5.1 Tasks

There are two main tasks in the SBY file:

- Task prove: Enable assertions and restrictions for validity tests.
- Task cover: Enable covers for satisfiability tests.

Running **sby -f axi_crossbar.sby cover | prove** will run SymbiYosys in either cover or prove mode depending on the argument chosen by the user. Running **sby -f axi_crossbar.sby** without arguments will run both cover and prove tasks at once.

5.2 Options and Engines

The section **options** has information to enable both prove and cover mode as shown below.

```
[options]
prove: mode prove
cover: mode cover
```

Figure 5: The SBY file options.

The selected engine for this example is **boolector**, and is defined as follows:

```
[engines]
smtbmc boolector
```

Figure 6: The SBY engine options.

5.3 Script

Finally, all the required files and the specific order is defined in the **script** section as follows:

```
[script]
# Read packages first
read -sv amba_axi4_protocol_checker_pkg.sv
read -sv amba_axi4_single_interface_requirements.sv
read -sv amba_axi4_definition_of_axi4_lite.sv
read -sv amba_axi4_atomic_accesses.sv
read -sv amba_axi4_transaction_structure.sv
read -sv amba_axi4_transaction_attributes.sv
read -sv amba_axi4_protocol_checker.sv

# Then the IP
read -incdir ../../../../src
read -sv amba_axi4_write_response_channel.sv
read -sv amba_axi4_write_address_channel.sv
read -sv amba_axi4_write_data_channel.sv
read -sv amba_axi4_read_data_channel.sv
read -sv amba_axi4_read_address_channel.sv
read -sv testbench.sv

# Then the crossbar sources
read -sv axi_crossbar_wr.v
read -sv axi_register_rd.v
read -sv arbiter.v
read -sv axi_crossbar_addr.v
read -sv axi_crossbar.v
read -sv axi_register_wr.v
read -sv priority_encoder.v
read -sv axi_crossbar_rd.v


# Report later: If I run without flatten, all properties with $rose fails
prep -flatten -top axi_crossbar
```

Figure 7: The SBY script options.

Note: The YosysHQ AXI4 sources are provided one by one in the SBY file, and not in a less verbose way such as include files, to be able to preview the source code in the SBY-GUI tool.

6 Running with SBY-GUI

It is recommended to use the sby-gui interface first to get familiar with the YosysHQ AXI4 Formal IP source organisation and the SBY tool. The Figure 8 shows this graphical interface that is executed with the

following command: **sby-gui <folder>** (in this case <folder> is **examples/axi_crossbar**). Executing the **play** button in any of the subwindows will execute the tasks defined in the upper left side.

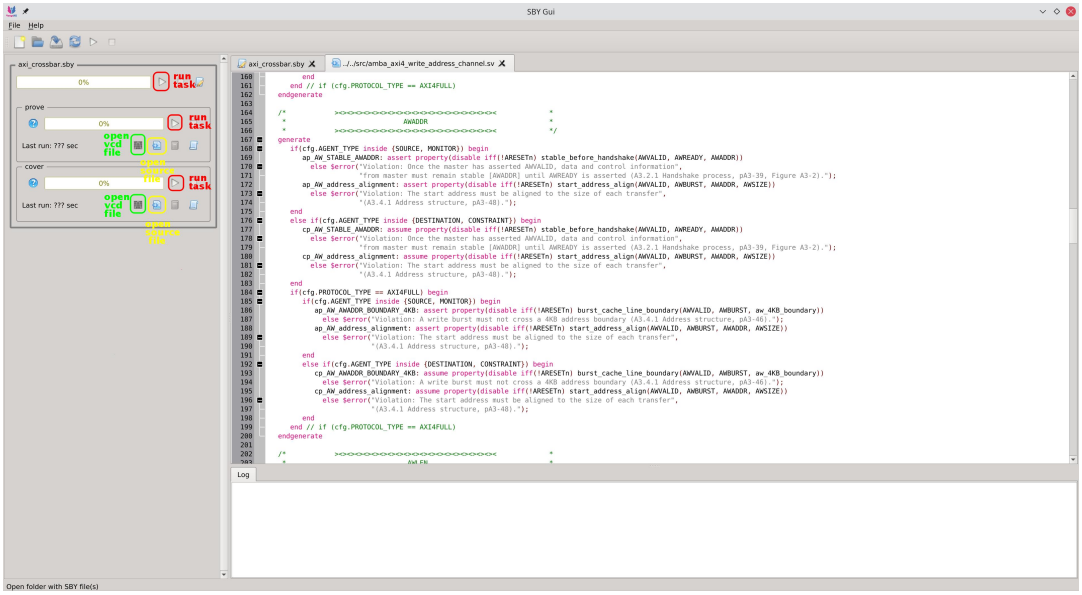

Figure 8: SBY Graphical Interface.

7 Running with Command Line

If command line is preferred, the example can be run with the following command: **sby -f axi_crossbar.sby cover | prove**. The user will need to monitor the terminal to check the output of the example.

Part IV
Errors

7.1 Unreachable Covers

There are 6 unreachable covers:

1. **source_chk.W_channel_checker.witness.genblk1.wp_WSTRB_NOT_ALL_VALID.**
2. **source_chk.B_channel_checker.witness.wp_BREADY_before_BVALID.**
3. **source_chk.AW_channel_checker.witness.wp_AW_B2B.**
4. **dest_check.AW_channel_checker.witness.wp_AW_NO_WAIT.**
5. **dest_check.AW_channel_checker.witness.wp_AWREADY_before_AWVALID.**
6. **dest_check.AR_channel_checker.witness.wp_ARREADY_before_ARVALID.**

7.2 Failed Assertions

7.2.1 Read burst crossing 4K address boundary.

The AXI4 Formal IP found a violation in the crossbar. Around time step 19, **ARBURST** = INCR, **ARLEN** = 1Ch, **ARSIZE** = 1h and **ARADDR** = 1EFE3h giving a final address of 1F01Bh, crossing the 4K boundary.

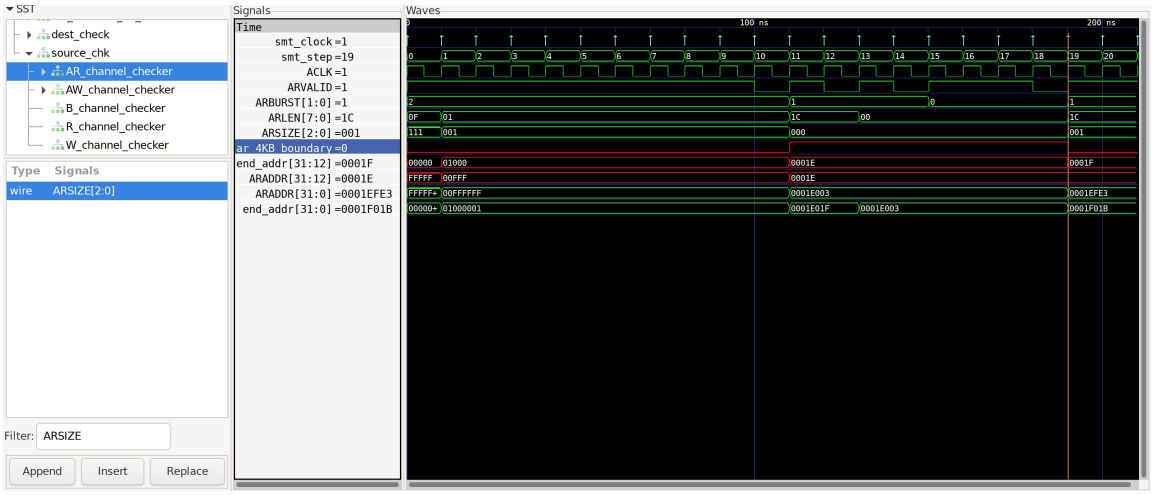

Figure 9: Assert failed in axi_crossbar: ap_AR_ARADDR_BOUNDARY_4KB.

Note: Signals are already configured to visualise in GTKWave by opening the file ap_AR_ARADDR_BOUNDARY_4KB.gtkw.

It is important to mention that this property also fails for AW channel.

Part V
Extending Checks

Appendix