Portland State
UNIVERSITY

ECE 593 Fundamentals of Pre-Silicon Functional Validation Project Verification Plan Group 7

Piyush Purwat (PSU ID – 941496418)

Venkata Sai Pranav Twarakavi (PSU ID – 951485914)

Mangala Bhat (PSU ID – 969559278)

Functional Specification:

https://opencores.org/projects/jtag

Portland State
UNIVERSITY

Verification Plan:

Technical requirements –

Description of verification levels

Unit level verification, using Mandatory tap ports to verify JTAG functionality

Functions to be verified:

The following instructions are verified.

1. EXTEST
2. SAMPLE_PRELOAD
3. IDCODE
4. DEBUG
5. BYPASS

Specific tests and methods –

Types of verification:

Greybox.

We are using this method as we keep the state transitions of the tap controller as our observation points.

Verification strategy

Portland State UNIVERSITY

Unit level,directed test cases for instructions and constrained randomization for other cases

Unit level constrained random testcases for instructions and randomized testcases for checking different scan chain combinations for different instructions by traversing the state machine to different states.

- Abstraction level

  Unit level

- Checking Strategy

Transaction based checking

Cycle based checking, the IO pins are bits and therefore output is checked every cycle using a reference model.

- **Coverage requirements**

We are targeting code coverage and functional coverage. Covergroups are to be written for the sixteen possible tap states and cross them with different output signals for different Instruction sequences.

Functional coverage is extracted for fsm transitions using internal variables that are asserted at each

Output state, additionally we use the variables used for decoding instruction and cross them with respective input to check whether

this combination occurs during the test and also check if relevant instruction is asserted for the given input

- Testing scenarios

1**. Reset**: The reset functionality is tested in three ways,
i. Asserting trst and checking state outputs
ii. Asserting tms with a sequence of 5 1's to achieve the same effect.
iii. Asserting tms with 5 1's/asserting trst in the middle of any instruction
2. BYPASS: We BYPASS chips to save test time. In this functional mode, after setting shiftDR, we load the IR with 1111 and check whether TDI is visible at TDO after a one clock cycle.
3. EXTEST: For this functional mode , we load the IR with 0000 , this mode connects the boundary scan chain and test data is driven off boundary outputs via boundary scan register to receive data via boundary inputs.
4. SAMPLE/PRELOAD: In this functional mode, the IR is loaded with 0001, we check whether we are able to collect functional data given to IC.Changes are observed in CaptureDR and ShiftDR state.
5. DEBUG: In this functional mode, IR mode is loaded with 1000 and enables debugging.
6. IDCODE: In this functional mode , we load IR with 0010.This allows required ID register to be selected .Additionally ID register should be available via a TAP data scan operation .Values updates across Capture,shift and Update states.
7. Random sequence with 2 instructions: Stimulus is given a constraint to provide two consecutive instructions with and without

reset.

8. Random sequence with all supported instruction: All instructions are applied once and compared with reference.

9. Random sequence with all instructions for multiple rounds: All instructions are applied for multiple rounds of operation in different orders with a reset in between each set of instructions. Project Management: -

single op:

Driving TMS and checking whether the design transitions to the

Required state for given sequence.

concurrent ops:

Driving both TMS and TDI to configure an instruction and drive data to data registers

sequences (back-to-back):

Driving multiple instructions and checking the ouput for the same

error cases:

Driving invalid instructions and checking if default conditions of design hold true

data dependent:

Apply constrained random stimulus and select bypass instruction and subsequently check whether input gets reflected in the output.

Portland State UNIVERSITY

Required tools :

1) EDA playground for sample code testing
2) QuestaSim for simulation
3) Linux Machine access through SSH - MobaXTerm
4) GITHub
5) Makefile

1) QuestaSim for simulation
2) Linux Machine access through SSH - MobaXTerm
3) GITHub
4) Makefile

- Risks and dependencies:

Questasim License

Less time available for completion of project

No designer support available since code is open source

- Resources requirements:

1) Comprehensive Functional Verification Verification: The complete industry cycle by Bruce Wile, John C. Gross, Wolfgang Roesner.

2) The Boundary Scan Handbook by Kenneth P. Parker

3) Materials from Prof. Tom Schubert.

4) JTAG Specification.

5) SystemVerilog LRM 2017 (1800-2017 - IEEE)

6) Design source code: taken from opencores.org –

Portland State UNIVERSITY

7.) People and resposibilities:

Piyush Purwat:

1.Developed constraints for relevant test cases,

2.Took part in debugging testbench issues

3.Developed classes like generator,driver.

4.Developed rtl designs like scan cell and circuit under test(ALU) (ongoing integration)

Mangala:

(1.Developed JTAG Reference model.

2.Took part in writing testcases

3.Contributed to debugging testbench issues.

4.Coding covergroups for coverage class

Venkata Sai Pranav

1.Contributed to debugging the testbench

2.Architectured basic testbench layout for the design.

3.Involved in verification planning,

4. Developed components like scoreboard and supported both coverage and testcase development

-Schedule details:

May 9 - Verification plan.

May 13 - Waveforms

May 20 - Project code drop1

Portland State
UNIVERSITY

May 27 - Project code drop2

June 3 - Final code drop

May 20th – Reference model creation

May 28th - Coverage model

June 2nd - Testcases and constraints

June 3rd - Debugging results