Verification Planning Mid Pro

Project Overview & scope

DUT Timer Summery

We are verifying a Timer DUT its main use is to countdown a programable Number loaded with automatic reload operation, the Timer control is achieved with req&gnt interface to control registers.

Verification Objectives

- Timing correctness
- Data integrity
- Functional accuracy behavior

In Scope

- Read/Write interface operation
- Timer behavior reset/reload_en

Out Scope

- Power consumption
- Other corner cases not mention

Open questions

- What is the internal register use?
  Do we need to check all the other option of the Timer?
- Does the START bit should posedge in order to restart the count?

Requirements Extraction & RTM

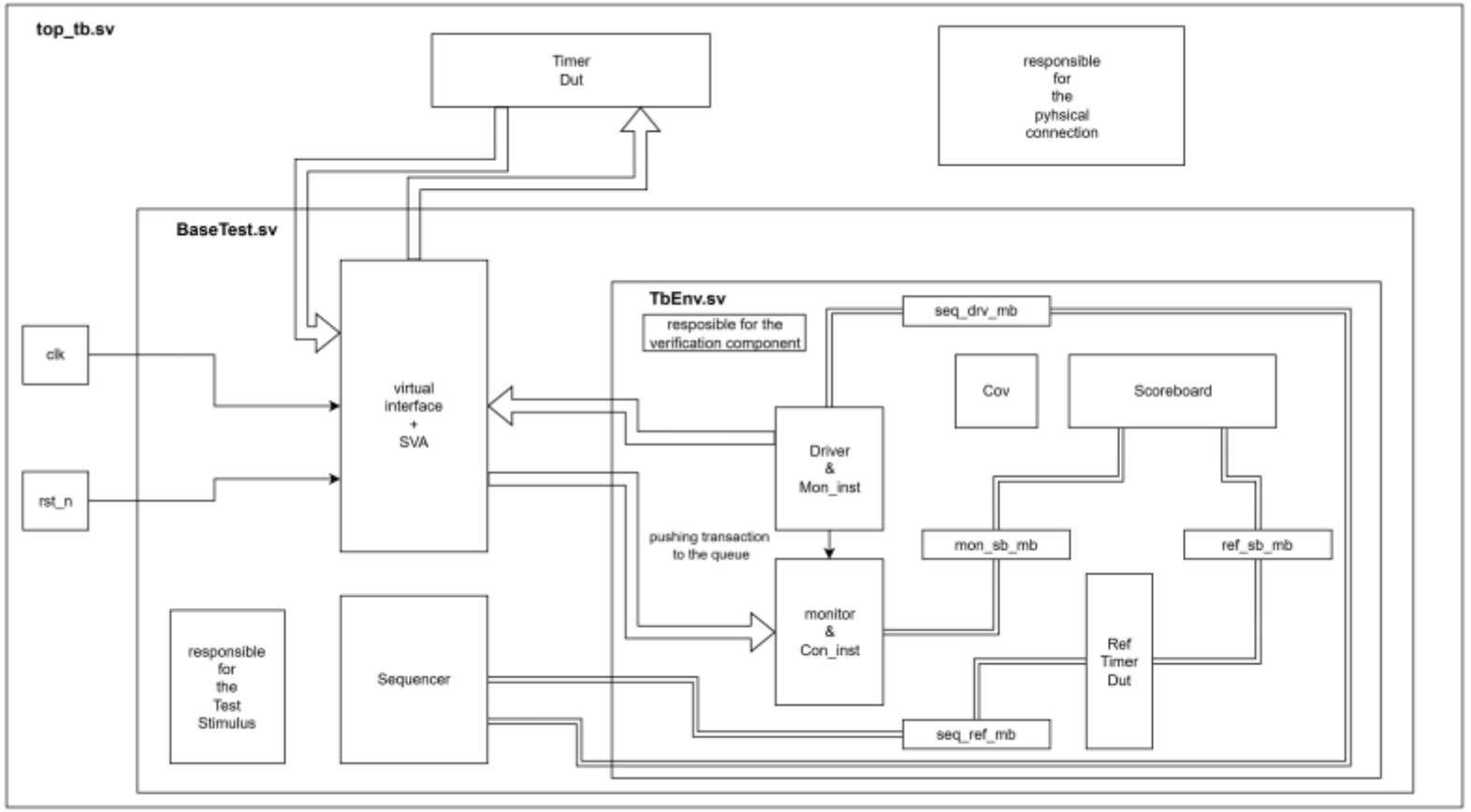
 # Metrics for Success (Exit

Req ID	Requirement	Verification Method	Planned Artifact
R1	Write mod: req -> gnt #3 clk	Assert	Assertion
R2	Read mod: Rdata -> gnt	Assert	Assertion
R3	Read/Write bus interface	coverage	Bus interface

Ve

Strategy Area	Approach
Methodology:	SystemVerilog object-oriented testbench architecture.
Stimulus Strategy:	Constrained-Random, with some Directed tests for corner cases (like Loading zero).
Checking Strategy:	Data checker for the expected results and SVA for the protocol operation
Coverage Strategy:	Coverage of the input/operation mode
Regression/Automation:	Creating a regression/automation baselined tests
Exit Criteria	Code Coverage (100% line/toggle) and Functional Coverage (100% of defined coverpoints).

Phase 3 – TB architecture



- Sequencer – creating the transactions according to our TB scenarioes
- REF_Model – our funvtional module that represent the behave of the DUT as we understand it from the designers.
- Driver – drives the transaction though the interface to the DUT.
- Monitor – getting the DUT outputs and convert it to transaction and pass it to the SB.
- Scoreboard – matching between the REF expected output and the actual output of the DUT.
- TB_top – connecting between the physical and the dynamic parts.
- Environment – gathering all the dynamic classes into one class as properties.
- BaseTest – making test scenarios by a sequence of the sequencer methods.

Phase 4 – covPlan & test scenarios

Timer
- START – countdown.
- Reload enable = high/low.
- Reset operation.

DUT commands
- Read/Write.
- Nothing.
- Unmapped address.

Cross
- Writing while reading mode.
- Reloading enable high while countdown.

Corner cases
- Loading zero.
- Start before countdown goes to zero.

Test name	Type	Purpose
Basic operation	Random	Loading valid data and str countdown should stop at the end.
Reset operation	directed	Should stop any operation and get to the ideal state
Reload_en	directed	Should state all over the countdown at the end with STR signal
Unmapped address	Random	Should continue without changing
WR while RD mode	directed	Should ignore the input data
Loading zero	directed	Should load 1 as default
Str while countdown	directed	Should str the countdown all over

Phase 5 – Stimulus Planning

Sequence name	description	Parameters / constraints
seq_load	Basic Write => read transaction	Addr within valid range, Wr_en
Seq_str_II	STR while countdown	

Phase 6 – Closure & Reporting strategy

Metric	Target	Justification
Code coverage		Why this level is sufficient
Functional coverage		Base on coverage goals
Assertion coverage		Which properties must trigger
Bug rate/ severity		Threshold for tape-out readiness

Reflection

- Which req was the hardest to make measurable?
- Where could randomization help u close coverage faster?
- What assumptions did u have to make about the DUT?