Asynchronous FIFO - Verification Plan

The goal of this verification plan is to ensure the Asynchronous FIFO transfers data between blocks with asynchronous clocks properly without metastability propagation. Key features to verify will include independent read/write clocks as well as FIFO full/empty detection. Binary to gray pointer conversion and synchronization across clock domains will also be important to verify. Directed tests will be used for corner cases along with constrained-random stimulus for stress testing. Assertions will be used to test clock domain crossing (CDC) safety.

Directed Tests

Test	Description
Reset	Reset read/write domains independently
Single read/write	Read one word, write one word
Fill FIFO	Fill until full
Drain FIFO	Drain until empty
Full condition	Write when full
Empty condition	Read when empty

Randomized Tests

- Random read/write block ready signals
- Random read/write clock frequencies
- Long random sequences with random reset assertions

Functional Coverage Plan

<table>
<tr><th colspan="2">Covergroups</th></tr>
<tr><td>Coverage Type</td><td>Behavior</td></tr>
<tr><td rowspan="3">Write-Side Coverage</td><td>Read block ready</td></tr>
<tr><td>Write when FIFO almost full</td></tr>
<tr><td>Write blocked when FIFO full</td></tr>
</table>

Read-Side Coverage	Write block ready
	Ready when FIFO almost empty
	Read blocked when FIFO empty
FIFO State Coverage	Empty <-> non-empty
	Full <-> non-full
	Simultaneous read/write
Pointer and CDC Coverage	Gray pointer transitions (single-bit change)
	Pointer synchronization latency (1 vs 2 cycles)
	Full/empty detection correctness across CDC

Code Coverage Goals

Metric	**Target**
Statement	100%
Branch	100%
Expression	>95%
Toggle	>95%
FSM	100%

Assertions

- No read/write when FIFO empty/full
- Gray pointers only increment one bit
- FIFO ordering preserved
- Pointer synchronization stability (no metastability glitches)

Team Member Responsibilities	
Carson	TB development, directed tests
David	Read/write driver, assertions, ordering checks
Liukee	Functional coverage, coverage analysis

Completion Schedule	
Milestone and Due Date	**Task**
M2-M3 (Feb 18th)	Develop TB skeleton and functional coverage model
	Implement directed test-cases and read/write drivers
	Validate interfaces and transactions
	Add ordering assertions
M4 (Feb 27th)	Define UVM architecture and tests
	Develop UVM driver/monitor, interfaces, and assertions
	Plan UVM coverage strategy and covergroups
M5 (Mar 14th)	Finalize UVM testcases and assertion validations
	Complete code/functional coverage and generate coverage reports
	Submission