Assumptions, Assertions and Covers:

Covers:

Statement	Description
bfm.AWVALID inside {0,1}	cover to check AWVALID signal only outputs valid values
bfm.AWREADY inside {0,1}	cover to check AWREADY signal only outputs valid values
bfm.WVALID inside {0,1}	cover to check WVALID signal only outputs valid values
bfm.WREADY inside {0,1}	cover to check WREADY signal only outputs valid values
bfm.BREADY inside {0,1}	cover to check BREADY signal only outputs valid values
bfm.BVALID inside {0,1}	cover to check BVALID signal only outputs valid values
bfm.ARVALID inside {0,1}	cover to check ARVALID signal only outputs valid values
bfm.ARREADY inside {0,1}	cover to check ARREADY signal only outputs valid values
bfm.RVALID inside {0,1}	cover to check RVALID signal only outputs valid values
bfm.RREADY inside {0,1}	cover to check RREADY signal only outputs valid values
rd_en == 1	cover to check rd_en goes high
wr_en == 1	cover to check wr_en goes high

Assumptions:

Statement	Description
$onehot({wr_en,rd_en})	Read and Write enable can't be high at the same time
(!$isunknown(Read_Address))	Read Address should be known
(!$isunknown(Write_Data))	Write Data should be known
($countbits(bfm.AWADDR) == 32)	Write Address should be 32 bits wide
($countbits(bfm.WDATA) == 32)	Write Data should be 32 bits wide
($countbits(bfm.RDATA) == 32)	Read Data should be 32 bits wide

Assertions:

Statement	Description	Pass /Fail
assertion1	AWADDR remains stable when AWVALID is asserted and AWREADY is low	Fail
assertion2	AWREADY is not asserted until AWVALID is high	Pass
assertion3	A value of X on AWADDR is not permitted when AWVALID is high	Pass
assertion4	AWVALID is LOW for the first cycle after reset signal goes high	Pass
assertion5	When AWVALID is asserted, then it remains asserted until AWREADY is high	Pass
assertion6	AWREADY is eventually true after AWVALID is asserted	Pass
assertion7	A value of X on AWVALID is not permitted when not in reset	Pass
assertion8	A value of X on AWREADY is not permitted when not in reset	Pass
assertion9	When AWVALID and AWREADY are high at the same time, the next cycle AWVALID goes low	Pass

assertion10	WDATA remains stable when WVALID is asserted and WREADY is low	Fail
assertion11	WREADY is not asserted until WVALID is high	Pass
assertion12	A value of X on WDATA is not permitted when WVALID is high	Pass
assertion13	WVALID is LOW for the first cycle after reset signal goes high	Pass
assertion14	When WVALID is asserted, then it remains asserted until WREADY is high	Pass
assertion15	WREADY is eventually true after WVALID is asserted	Pass
assertion16	A value of X on WVALID is not permitted when not in reset	Pass
assertion17	A value of X on WREADY is not permitted when not in reset	Pass
assertion18	When WVALID and WREADY are high at the same time, the next cycle AWVALID is low	Pass
assertion19	When AWVALID is high and AWREADY is low, BVALID is low until AWREADY is high	Pass
assertion20	When WVALID and WREADY are high BVALID and BREADY will be high	Pass
assertion21	A value of X on BVALID is not permitted when not in reset	Pass
assertion22	A value of X on BREADY is not permitted when not in reset	Pass
assertion23	BREADY is eventually true after BVALID is asserted	Pass
assertion24	When BVALID is asserted, then it must remain asserted until BREADY is HIGH	Pass
assertion25	ARADDR remains stable when ARVALID is asserted and ARREADY is low	Fail
assertion26	A value of X on ARADDR is not permitted when ARVALID is high	Pass
assertion27	ARVALID is LOW for the first cycle after reset goes high	Pass
assertion28	When ARVALID is asserted, then it remains asserted until ARREADY is high	Pass
assertion29	A value of X on ARVALID is not permitted when not in reset	Pass
assertion30	A value of X on ARREADY is not permitted when not in reset	Pass
assertion31	ARREADY is eventually true after ARVALID is asserted	Pass
assertion32	When ARVALID and ARREADY are high at the same time, the next cycle ARVALID goes low	Pass
assertion33	RDATA remains stable when RVALID is asserted, and RREADY is low	Fail
assertion34	A value of X on RDATA valid byte lanes is not permitted when RVALID is high.	Pass
assertion35	RVALID is LOW for the first cycle after reset goes high	Pass
assertion36	When RVALID is asserted, then it must remain asserted until RREADY is high	Pass

assertion37	A value of X on RVALID is not permitted when not in reset	Pass
assertion38	A value of X on RREADY is not permitted when not in reset	Pass
assertion39	RREADY is eventually true after RVALID is asserted	Pass
assertion40	When RVALID and RREADY are high at the same time, the next cycle RVALID goes low	Pass
assertion41	When AWVALID and AWREADY are high, WVALID and WREADY are high eventually	Pass
assertion42	When ARVALID and ARREADY are high, RVALID and RREADY are high eventually	Pass
assertion43	ARREADY is not asserted until ARVALID goes high	Pass
assertion44	AWREADY is not asserted until AWVALID goes high	Pass
assertion45	WREADY is not asserted until WVALID goes high	Pass
assertion46	BVALID is not asserted until AWREADY goes high	Pass
assertion47	BVALID is not asserted until WREADY goes high	Pass
assertion48	WREADY is not asserted until WVALID and AWVALID is asserted	Pass

Bugs Encountered:

After writing multiple assertions and validating the design, we discovered a few bugs during execution with the FPV tool. Upon debugging and backtracking, it became apparent that the AXI4 Lite design we initially worked with had inherent flaws.

Property	Description of Bug
(bfm.ARVALID) && (!bfm.ARREADY) |=> $stable(bfm.ARADDR) && (bfm.ARVALID)	Read Address Remains Stable when Read Address Valid is asserted and Read Address Ready is low
(bfm.RVALID) && (!bfm.RREADY) |=> $stable(bfm.RDATA) && (bfm.RVALID)	Read Data Remains Stable when Read Data Valid is asserted and Read Data Ready is low
(bfm.AWVALID && !bfm.AWREADY) |=> $stable(bfm.AWADDR) && (bfm.AWVALID)	Write Address Remains Stable when Write Address Valid is asserted and Write Address Ready is low
(bfm.WVALID && (!bfm.WREADY)) |=> $stable(bfm.WDATA) && bfm.WVALID	Write Data Remains Stable when Write Data Valid is asserted and Write Data Ready is low