verification plan for top level

Module Tested	Feature Tested	Test Name	Criteria Number	Description of Test Inputs	Description of Expected Outputs
SoC Core	Normal pre-inference behavior	Normal Operation Before Inference	1	Run the SoC with basic operations (reads/writes/arithmetic) without interacting with the accelerator.	SoC runs normally with no unintended accelerator access. Accelerator status remains in idle state.
SoC → Subordinate	Weight write sequence	Transfer Weights to Subordinate	2	SoC performs a full series of bus writes to the accelerator's weight address space using known weight patterns.	Accelerator accepts all writes, stores weights in exact order, and triggers any internal "weights available" state if present.
Accelerator	Load weights into systolic array	Load Weights into Array	3	After writing weights, SoC sets the "load weights" control bit. No new writes occur during loading.	Accelerator enters a loading phase: busy asserted, weights moved into systolic array, and "weights loaded" becomes true once finished.
SoC ↔ Accelerator	Polling weight load status	Poll Weight Load Completion	4	SoC repeatedly reads accelerator status after issuing the load command.	During loading: status shows weights_not_ready. After completion: the SoC eventually reads weights_ready = 1. No incorrect early-ready signals.
SoC → Subordinate	Input vector transfer	Send Input Vectors	5	SoC writes multiple input vectors with randomized values to the accelerator's input memory region.	Accelerator stores all inputs in correct order; any input-ready counter or validity flag updates properly. No dropped or corrupted values.
SoC → Accelerator	Start inference command	Start Inference	6	After all inputs are written, SoC writes the "start inference" command to the accelerator.	Accelerator transitions to compute mode; busy or inference_active becomes 1. No computation starts early before the command.
Accelerator	Execute full inference	Run Inference Operation	7	Accelerator runs systolic array computation using the previously loaded weights and inputs.	Computation completes successfully; internal array produces correct outputs; accelerator stores results in output buffer and clears busy.
SoC ↔ Accelerator	Poll inference completion	Poll Inference Done	8	While the accelerator is running, SoC repeatedly reads its status register until completion.	During compute: busy=1, done=0. After computation: busy=0, done=1. Done signal remains stable until software clears it (if defined).
SoC ↔ Subordinate	Output retrieval	Retrieve Output Data	9	After detecting done=1, the SoC reads all outputs from the accelerator's output register space and compares to a golden model.	All outputs match expected inference results with no missing, extra, or corrupted values. Output-valid indicators update correctly as data is read.

AHB Subordinate Verification Plan

Module Tested	Feature Tested	Test Name	Criteria Number	Description of Test Inputs	Description of Expected Outputs
AHB Subordinate	Address decode & valid write	Write weight buffer (all bytes)	N/A	Send write transfers to addresses 00–07. Try different transfer sizes like byte, half-word, word, and full 64-bit. Use simple data patterns so I can easily check the stored value.	The write should be accepted with no error. The stored data in the weight buffer should match what was written, and the correct byte or word lane should update.
AHB Subordinate	Address decode & valid write	Write input buffer (all bytes)	N/A	Same idea as the weight test, but write to addresses 08–0F. Use different data patterns to make it clear this is input data.	Writes should go through with no error. Input buffer should update with the correct data according to the address and transfer size.
AHB Subordinate	Bias register read/write	Bias register full-width access at 10	N/A	Write a full 64-bit value to address 10. Then read it back several times using byte, half-word, word, and full 64-bit transfers.	All reads should return the correct part of the stored 64-bit bias value. No error should happen, and the data should not shift or get corrupted.
AHB Subordinate	Bias register sliced reads	Bias byte and half-word reads at 11–17	N/A	Preload the bias register with a known pattern. Then read from addresses 11–17 which each map to different byte positions. Test the allowed transfer sizes for each address.	The returned data should match the specific byte or half-word for that address. Upper bits should be zero, and no error should be raised.
AHB Subordinate	Output register read	Output read at 18–1F	N/A	Force the output register to a fixed value. Do reads on addresses 18–1F with different transfer sizes so each slice is checked.	The read data should show the correct output slice depending on the address. No errors should appear, and data truncation should follow the design.
AHB Subordinate	Error register read	Error read at 20–21	N/A	Set the internal error bits for buffer occupancy and device busy. Then read from addresses 20 and 21 with allowed sizes.	The read value should match the current error bits. No extra bits should be set, and no write should be allowed here.
AHB Subordinate	Control register read/write	Control register test at 22	N/A	Write different combinations to the control register (bit 0 start, bit 1 load). After each write, perform a read to make sure the value updates.	Control register should store the last written value. Reads should show the correct bit pattern, and no error should occur.
AHB Subordinate	Status register read	Status read at 23	N/A	Change the internal status bits such as inference done and busy. Then read from address 23.	The read value should reflect the current status bits exactly. No writes should affect this register.

Module Tested	Feature Tested	Test Name	Criteria Number	Description of Test Inputs	Description of Expected Outputs
AHB Subordinate	RAW hazard same address	Write then read same address	N/A	Perform a write to address 10, 22, or 24. Immediately next cycle, do a read to the same address with the same transfer size.	The read should show the new written value right away, not the old one. No errors should happen.
AHB Subordinate	RAW hazard sliced reads	Write full 64-bit then read slices	N/A	Write a 64-bit pattern to address 10. In the next cycle, read from the slice addresses 11–17.	All slice reads should show the updated data, meaning there is no hazard or delay.
AHB Subordinate	RAW hazard across registers	Back-to-back to different registers	N/A	Write to one register (like 22), then quickly read another register (like 23) without delay.	Reads should return the correct and current value of the second register. No mixing or corruption across registers.

verification plan for controller & sram

Module Tested	Feature Tested	Test Name	Criteria Number	Description of Test Inputs	Description of Expected Outputs
sram_ controller	load in data	load in weight data	N/A	AHB subordinate sends the write weight signal. Sram can't not be full	1.the weight data should be stored into the correct address without any error happen 2.write index should increase by 2 per weight
sram_ controller	load in data	load in input data	N/A	AHB subordinate sends the write input signal. Sram can't not be full	1.the input data should be stored into the correct address without any error happen 2.write index should increase by 2 per input
sram_ controller	load in data	load in activations	N/A	AHB subordinate sends the write input signal. Sram can't not be full	1. the output data should be stored into the correct address without any error happen 2.write index should increase by 2 per output
sram_ controller	read data out	read weight data	N/A	AHB subordinate sends the load weight signal. Sram can't not be empty	1.sram read data should match the previous write data with the same address. 2.read index should increase by 2 per weight 3.8 cycles is required so 64 bytes of weight are sent
sram_ controller	read data out	read input	N/A	AHB subordinate sends the start inference signal. Sram can't not be empty	1.sram read data should match the previous write data with the same address. 2.read index should increase by 2 per weight 3.8 cycles is required so 64 bytes of weight are sent
sram_ controller	read data out	read output	N/A	AHB subordinate sends the output read signal. Sram can't not be empty	1.sram read data should match the previous write data with the same address. 2.read index should increase by 2 per weight 3.8 cycles is required so 64 bytes of weight are sent
sram_ controller	occupancy error	read empty test	N/A	For input, weight, output buffer read data from sram when read, write count is the same and read flag is the same as write flag	1.occupancy error should rise 2.depends on different kind of data raise the corresponding flag
sram_ controller	occuoancy error	write full test	N/A	For input, weight, output buffer write new data into it when read, write count is the same and read flag different then write flag	1.occupancy error should rise 2.depends on different kind of data raise the corresponding flag
sram_ controller	device error	device busy error test	N/A	In all design busy states when read write to sram command is asserted	1.device busy error raise
sram_ controller	sram error	sram error test	N/A	when data is processing but sram state goes to error	1.goes back to idle ready for new transaction

verification plan for 8x8systolic_array&&bias_adder&&activations

Module Tested	Feature Tested	Test Name	Criteria #	Description of Test Inputs	Description of Expected Outputs
systolic_array_core	reset behavior	reset test	N/A	Hold n_rst low for 2 cycles while all inputs stay idle; then release reset and keep system idle.	All outputs and flags remain 0; no X/Z states; system enters clean ready state after reset.
systolic_array_core	MAC operation	pattern weights small input	N/A	Preload weights with w_reg[r][c] = r + c; send 8 small input vectors using send_batch.	array_busy asserts during compute; activations_v asserts when outputs become ready; all 8 outputs match expected MAC results; overflow_flag remains 0.
systolic_array_core	stall handling	design_busy stall test	N/A	Assert design_busy for 3 cycles during input streaming while input vectors continue driving.	Pipeline stalls cleanly with no data corruption; outputs remain correct; activations_v and overflow_flag do not glitch.
systolic_array_core	overflow detection	clipping test	N/A	Set all weights = +64; send input vectors of all +64 using send_batch.	MAC results exceed range and clip; overflow_flag asserts; outputs show saturated (clipped) values.
bias_adder	bias addition	basic bias test	N/A	outputs holds small values; bias = +1 on all lanes; activations_v = 1.	Output equals outputs + bias; no overflow; no glitches during the valid cycle.

bias_adder	overflow behavior	large bias test	N/A	outputs hold values near ±127; bias = ±20; activations_v = 1.	Output saturates or wraps based on design rule; no invalid X/Z values appear.
bias_adder	enable gating	activations_v test	N/A	One cycle with activations_v = 1, then activations_v = 0 while input data continues changing.	Output updates only on valid cycle; output remains stable when activations_v = 0.
activations	ReLU	relu mode test	N/A	sum_out has mixed pos/neg values; activation_mode = 000.	Positive values pass unchanged; negative values become 0.
activations	binary activation	binary mode test	N/A	sum_out contains mixed values; activation_mode = 001.	Non-negative values → +1; negative values → −1.
activations	identity	identity mode test	N/A	sum_out contains mixed values; activation_mode = 010.	Output equals input value for all lanes.
activations	leaky relu	leaky relu test	N/A	Negative values present; activation_mode = 011.	Positive values unchanged; negative values multiplied by leak factor.