Module Name: Design Verification Onboarding

Design Engineers: Sarvajith K

Verification Engineers: Sarvajith K

Test Bench File:

Functionality	Basic Computation (write to low SRAM addresses 0x00 & 0x01)
☐ Normal addition with no overflow	Add just two small numbers; A =2, B = 3 -> A + B A+B = Result = 5, no overflow
☐ Normal Addition with overflow	Add two max values: A = 0xFFFFFFFF, B = 0x1 → A + B = 0x00000000 (wraps to 0), overflow flag = 1.

Functionality	Edge Cases write to (0x02 & 0x03)
☐ Test extreme input	Zero addition: 0+0 -> 0 Max + 0 -> Max Max = 0xFFFFFFFF

Functionality	Data Flow and SRAM Integrity (write to 0x10 and 0x11)
☐ Data Read	Write a value to SRAM, then read back. Write 0x1234 to address 0x10 -> Read at 0x10 returns 0x1234
☐ Data Write	Overwrite existing value in SRAM -> Write 0x5678 to address 0x10 -> read at 0x10 returns 0x5678

Functionality	Timing & Reset Tests (Assertion)
☐ Reset functionality Read from 0x20-0x22 and write to 0x30-0x32	Apply reset while funcitoning - > Reset = 1 after some addition -> DUT output resets to idle/0 state. assert property @(posedge clk) calc_if.reset |-> ##1 (state==S_IDLE);
☐ Verify the buffer_loc toggling logic.	assert property (@(posedge clk) (!calc_if.reset && state == S_RWAIT) |-> ##1 (my_calc.u_ctrl.buffer_control_q == $past(my_calc.u_ctrl.half));

☐ Verify address limit	assert property @(posedge clk)(!calc_if.reset && (my_calc.read || my_calc.write)) |-> ((my_calc.r_addr < 512) && (my_calc.w_addr < 512));
☐ Verify start address ordering	assert property @(posedge clk)(!calc_if.reset && state != S_IDLE && state != S_END) |->((calc_if.read_start_addr <= calc_if.read_end_addr) && (calc_if.write_start_addr <= calc_if.write_end_addr));
☐ Verify start/end ordering for **read addresses**	assert property @(posedge clk)(!calc_if.reset && my_calc.read) |-> (my_calc.r_addr >= calc_if.read_start_addr && my_calc.r_addr <= calc_if.read_end_addr);
☐ Verify start/end ordering for **write addresses**	assert property @(posedge clk)(!calc_if.reset && my_calc.write) |-> (my_calc.w_addr >= calc_if.write_start_addr && my_calc.w_addr <= calc_if.write_end_addr);

Functionality	Random Constrained Test (Coverage)
☐ give random input	Verify the output. Overall Coverage for DUT >= 96%

Functionality	FSM Coverage
☐ FSM coverage	Coverage reaches 100%