MULTI-MODE ALU WITH INTERRUPT SUPPORT

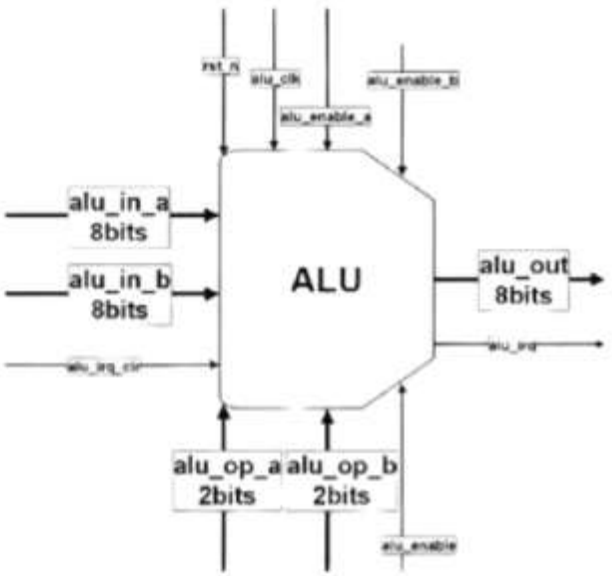
Group 2
Eng. Youssef Nasser
Eng. Ahmed Ashraf
Eng. Ahmed Ehab
Eng. Abdulrahman Sherif

AGENDA

1. Design Requirements
2. Verification Plan
3. Environment Architecture
4. Makefile
5. Simulation Results on VCS
6. Coverage Plan on Verdi
7. Resources

DESIGN REQUIREMENTS

DESIGN REQUIREMENTS



Operation a == alu_enable_a && alu_enable				
alu_op_a	alu_out	alu_in_a	alu_in_b	alu_irq == 1'b1
2'b00	AND	ALL	!=8'h0	alu_out == 8'hFF
2'b01	NAND	!=8'hFF	!=8'h03	alu_out == 8'h00
2'b10	OR	ALL	ALL	alu_out == 8'hF8
2'b11	XOR	ALL	ALL	alu_out == 8'h83

Operation b == alu_enable_b && alu_enable				
alu_op_b	alu_out	alu_in_a	alu_in_b	alu_irq == 1'b1
2'b00	XNOR	ALL	ALL	alu_out == 8'hF1
2'b01	AND	ALL	!=8'h03	alu_out == 8'hF4
2'b10	NOR	!=8'hF5	ALL	alu_out == 8'hF5
2'b11	OR	ALL	ALL	alu_out == 8'hFF

DESIGN REQUIREMENTS

ID	Signal	Description
DR1	alu_clk	• Frequency is 32 MHz (31 nanoseconds 1000/32) • Positive edge triggered clock.

DESIGN REQUIREMENTS

ID	Signal	Description
DR2	rst_n	• Active low, asynchronous reset. • Whenever rst_n is low, then alu_out and alu_irq should be driven to 0. • rst_n should be asserted at least once at start-up.

DESIGN REQUIREMENTS

ID	Signal	Description
DR3	alu_in_a	• Synchronized to alu_clk, positive edge trigger. • When alu_op_a == NAND , alu_enable == 1, alu_enable_a == 1 and alu_enable_b == 0 then alu_in_a can’t take value of 8’hff. • When alu_op_b == NOR, alu_enable == 1, alu_enable_a == 0 and alu_enable_b == 1 then alu_in_a can’t take a value of 8’hf5.

DESIGN REQUIREMENTS

ID	Signal	Description
DR4	alu_in_b	• Synchronized to alu_clk, positive edge trigger. • When alu_op_a == AND, alu_enable == 1, alu_enable_a == 1 and alu_enable_b == 0 then alu_in_b can’t take value of 8’h0. • When alu_op_a == NAND, alu_enable == 1, alu_enable_a == 1 and alu_enable_b == 0 then alu_in_b can’t take value of 8’h03. • When alu_op_a == AND, alu_enable == 1, alu_enable_a == 0 and alu_enable_b == 1 then alu_in_b can’t take value of 8’h03.

DESIGN REQUIREMENTS

ID	Signal	Description
DR5	alu_enable	• Synchronized to alu_clk, positive edge trigger. • If alu_enable is de-asserted: • All outputs should be maintained, if rst_n is high, regardless of alu_enable_a or alu_enable_b. • If alu_enable is asserted: • alu_enable_a and alu_enable_b can’t be asserted at the same time.

DESIGN REQUIREMENTS

ID	Signal	Description
DR6	• alu_enable_a • alu_enable_b	• Synchronized to alu_clk, positive edge trigger. • When alu_enable is asserted: • if alu_enable_a == 0 and alu_enable_b == 0, it is legal and all outputs should be maintained. • if alu_enable_a == 0 and alu_enable_b == 1, it is legal and the ALU should do operation b. • if alu_enable_a == 1 and alu_enable_b == 0, it is legal and the ALU should do operation a. • if alu_enable_a == 1 and alu_enable_b == 1, it is illegal and the output depends on RTL that cover this case. (Note: RTL can have //full case Synopsys so this case will be excluded.)

DESIGN REQUIREMENTS

ID	Signal	Description
DR7	alu_op_a	• Synchronized to alu_clk, positive edge trigger. • When it is 2'b00, alu_out should be the AND-ing of the inputs. • When it is 2'b01, alu_out should be the NAND-ing of the inputs. • When it is 2'b10, alu_out should be the OR-ing of the inputs. • When it is 2'b11, alu_out should be the XOR-ing of the inputs. • (Note: The design may don't have a default case, all cases are covered)

DESIGN REQUIREMENTS

ID	Signal	Description
DR8	alu_op_b	• Synchronized to alu_clk, positive edge trigger. • When it is 2'b00, alu_out should be the XNOR-ing of the inputs. • When it is 2'b01, alu_out should be the AND-ing of the inputs. • When it is 2'b10, alu_out should be the NOR-ing of the inputs. • When it is 2'b11, alu_out should be the OR-ing of the inputs. • (Note: The design may don't have a default case, all cases are covered)

DESIGN REQUIREMENTS

ID	Signal	Description
DR9	alu_irq_clr	• Synchronized to alu_clk, positive edge trigger. • It should be high for at least one clock cycle then it gets de-asserted. • It should be high after at least one clock cycle from the assertion of alu_irq. • It is an active high clear signal.

DESIGN REQUIREMENTS

ID	Signal	Description
DR10	alu_out	• Synchronized to alu_clk, positive edge trigger. • Should be as expected, according to the given table, after one clock cycle from applying the inputs. • If alu_enable and alu_enable_a are asserted, and alu_enable b is de-asserted while alu_op_a is NAND then alu_out never be 8'h00, due to the constrain on alu_in_a.

DESIGN REQUIREMENTS

ID	Signal	Description
DR11	alu_irq	• Synchronized to alu_clk, positive edge trigger. • It should be asserted after one clock cycle from the event that triggers it. • Events that assert it to high, in case of other events alu_irq maintains its value.

DESIGN REQUIREMENTS

ID	Signal	Description
DR11	alu_irq	• When alu_enable && alu_enable_a && !alu_enable b, operation a: ○ If alu_op_a is AND, alu_in_a is 8'hFF and alu_in_b is 8'hFF. ○ If alu_op_a is NAND and ~(in_a & in b) equals 8'h00. (Note: this case can't happen) ○ If alu_op_a is OR and (alu_in_a | alu_in_b) equals 8'hE8. ○ If alu_op_a is XOR and (alu_in_a ^ alu_in_b) equals 8'h83.

DESIGN REQUIREMENTS

ID	Signal	Description
DR11	alu_irq	• When alu_enable && alu_enable_b && !alu_enable a, operation b: ○ If alu_op_b is XNOR and (alu_in_a ~^ alu_in_b) equals 8'hF1. ○ If alu_op_b is AND and (alu_in_a & alu_in b) equals 8'hF4. ○ If alu_op_b is NOR and ~(alu_in_a | alu_in_b) equals 8'hF5. ○ If alu_op_b is OR and (alu_in_a | alu_in_b) equals 8'hFF.

DESIGN REQUIREMENTS

ID	Signal	Description
DR11	alu_irq	• It should be high until alu_irq_clr is asserted. • It should be de-asserted after one clock cycle from asserting alu_irq_clr. • Event has a priority over alu_irq_clr, when there is an event from the previous ones and alu_irq_clr is asserted at that time, then alu_irq should be high.

VERIFICATION PLAN

VERIFICATION PLAN

ID	Signal	Description
VR1	alu_clk	Driven positive edge trigger clock with a period of 31.25 ns, (1000/32MHz). (Note: timescale 1ns/1ps)

VERIFICATION PLAN

ID	Signal	Description
VR2	rst_n	• Drive it with 0 at the start, then it can be 0 or 1 at different times during normal operation independent of the clock, most of the time it is high. • Cover that reset is de-asserted and asserted. • Check that outputs are driven to all 0's immediately when it is low. • Check that outputs are as expected when it is high.

VERIFICATION PLAN

ID	Signal	Description
VR3	alu_in_a	• Driven constrained randomly. • Cover that the forbidden values aren’t generated. • Cover that all input pins toggle from 1 to 0 and vice versa.

VERIFICATION PLAN

ID	Signal	Description
VR4	alu_in_b	• Driven constrained randomly. • Cover that the forbidden values are not generated. • Cover that all input pins toggle from 1 to 0 and vice versa.

VERIFICATION PLAN

ID	Signal	Description
VR5	alu_enable	• Drive it with 1 or 0, most of the time it is high. • Cover that alu_enable, alu_enable_a and alu_enable_b toggle from 1 to 0 and vice versa • Check that all outputs are maintained when alu_enable is low.

VERIFICATION PLAN

ID	Signal	Description
VR6	alu_enable_a	• Drive it with 1 or 0, most of the time alu_enable_a dosen’t equal alu_enable_b. • Cover that alu_enable_a toggles with all variations of alu_enable. • Check that when alu_enable_a is high, alu_out is generated according to operation a.

VERIFICATION PLAN

ID	Signal	Description
VR7	alu_enable_b	• Drive it with 1 or 0, most of the time alu_enable_a dosen't equal alu_enable_b. • Cover that alu_enable_b toggles with all variations of alu_enable. • Check that when alu_enable_b is high, alu_out is generated according to operation b

VERIFICATION PLAN

ID	Signal	Description
VR7	alu_op_a	• Driven with all possible combinations randomly. • Cover all possible combinations while alu_enable and alu_enable_a are high, and cover all other possible cases, listed in design requirements. • Check that when the operation is NAND while alu_enable and alu_enable_a are asserted to high, alu_out will never be 8’h00.

VERIFICATION PLAN

ID	Signal	Description
VR8	alu_op_b	• Driven with all possible combinations randomly. • Cover all possible combinations while alu_enable and alu_enable_b are high, and cover all other possible cases, listed in design requirements.

VERIFICATION PLAN

ID	Signal	Description
VR9	alu_irq_clr	• Drive it with 1 while alu_irq is de-asserted. • Cover alu_irq_clr that is asserted and alu_irq is de-asserted. • Check if alu_irq is still 0 or not. • Drive it with 1 after alu_irq is asserted. • Cover that alu_irq_clr is asserted and alu_irq is also asserted. • Check that alu_irq is cleared, equals 0. • Drive it with 0 regardless of the value of alu_irq, 1 or 0. • Cover that alu_irq_clr is low when alu_irq is asserted and de-asserted at different times. • Check that alu_irq is maintained.

VERIFICATION PLAN

ID	Signal	Description
VR10	alu_out	• Cover that all alu_out combinations that asserts alu_irq are generated. • Cover that alu_out pins toggles. • Check that alu_out is asserted after one clock cycle from applying the inputs. • Check that alu_out is maintained when alu_enable_a and alu_enable_b are asserted to low at the same time.

VERIFICATION PLAN

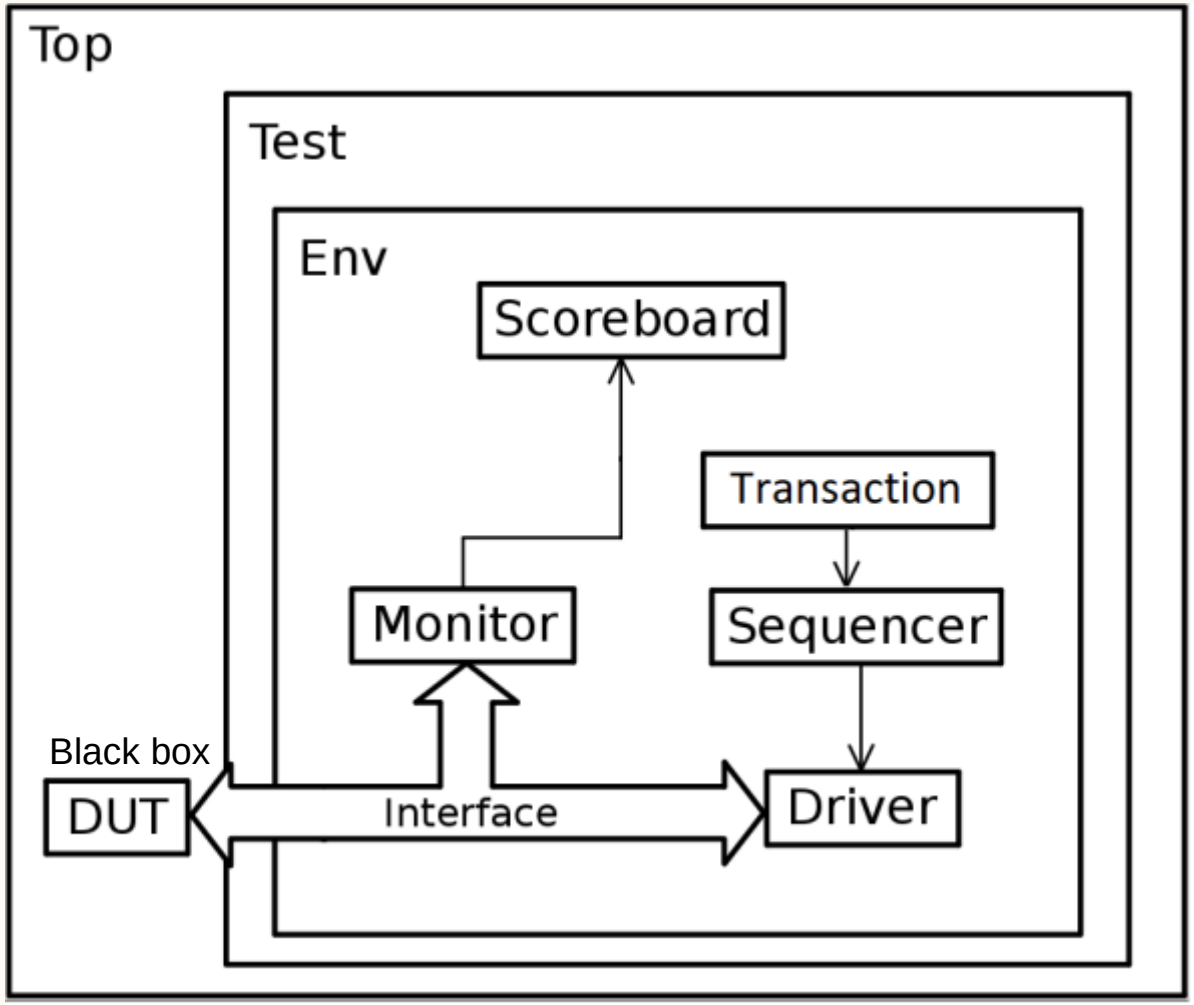
ID	Signal	Description
VR11	alu_irq	• Check that alu_irq is asserted to high at the same cycle as alu_out. • Check that alu_irq isn't asserted unless alu_out is one of the cases listed in the design requirements. • Drive alu_irq_clr to high while there are two successive events that assert alu_irq.

VERIFICATION PLAN

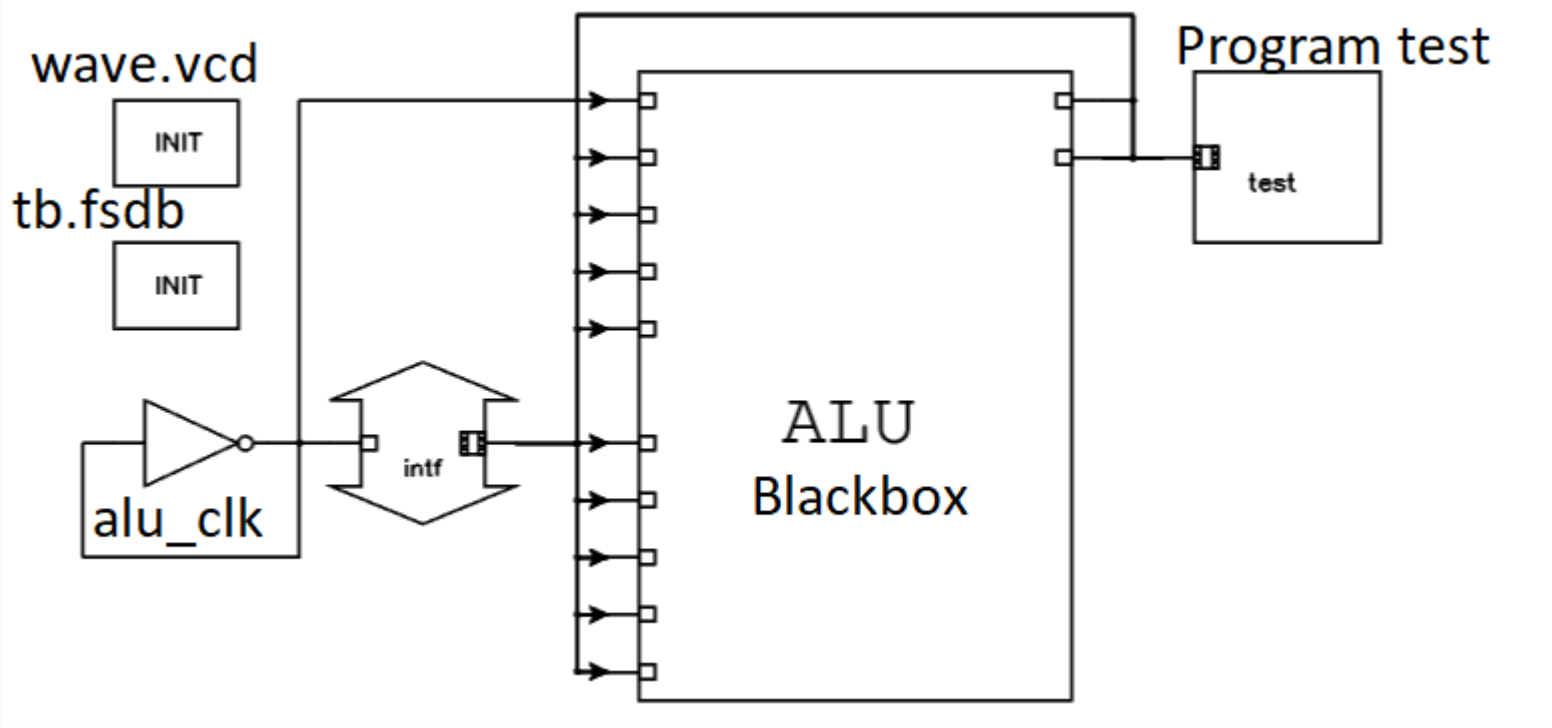
ID	Signal	Description
VR11	alu_irq	• Cover that alu_irq_clr is asserted while there is two successive events that assert alu_irq. • Check that events have priority over alu_irq_clr • Check that alu_irq is low when alu_irq_clr high, taking into consideration that there isn't two successive events. • Check that alu_irq is still asserted when there is two successive events while alu_irq_clr is high. • Check that alu_irq is de-asserted after one clock cycle from the assertion of alu_irq_clr.

ENVIRONMENT ARCHITECTURE

ENVIRONMENT ARCHITECTURE



ENVIRONMENT ARCHITECTURE



MAKEFILE

MAKEFILE

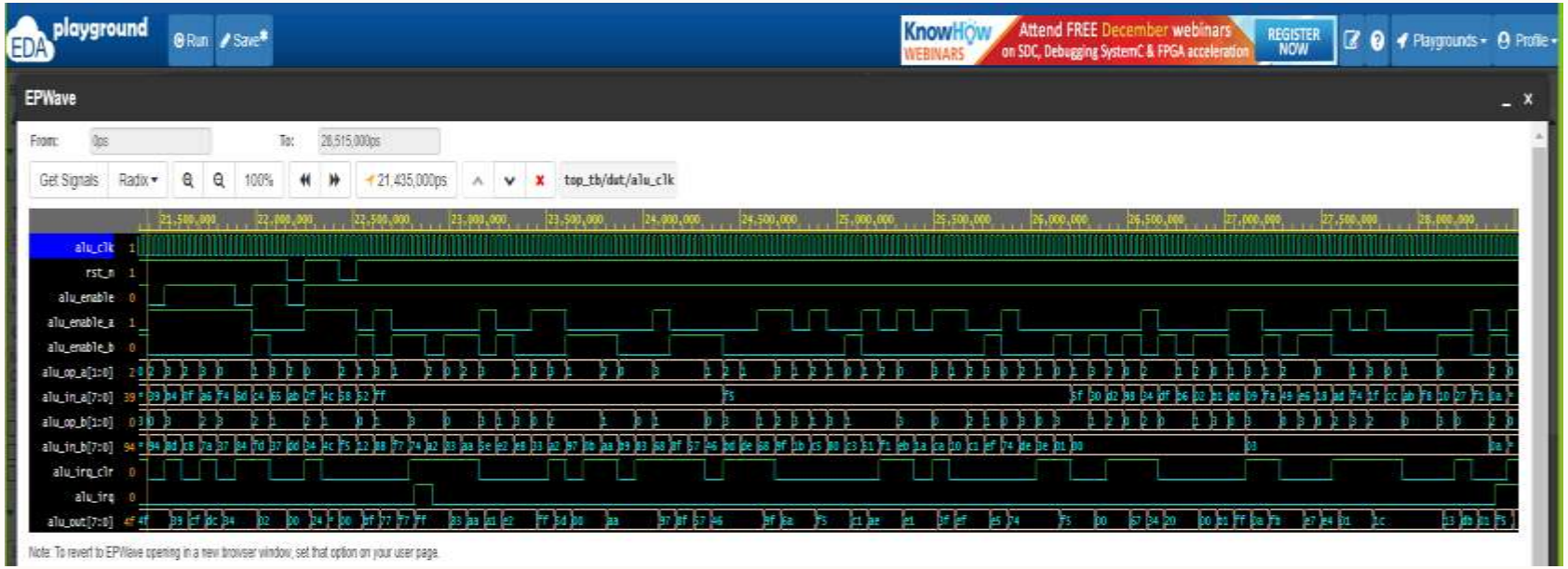
```
#-------------------------------------------
comp  : clean vcs coverage_html cov_verdi
#-------------------------------------------
vcs   :
    vcs  \
              -f files.f -cm line+fsm+tgl+branch+cond \
              -timescale=1ns/1ps +vcs+flush+all \
              -fsdb  -full64  -R  +vc  +v2k  -sverilog  -debug_all  \
              -gui -l vcs_out.log &
#-------------------------------------
verdi  :
    verdi -f files.f -ssf -sv tb.fsdb &
#---------------------------------
coverage_text  :
    urg -dir simv.vdb -format text &
#-------------------------------
coverage_html  :
    urg -dir simv.vdb &
#-----------------------------------
cov_verdi  :
    verdi -cov -covdir simv.vdb &
#-----------------------------------------
clean  :
     rm  -rf  *~  core  csrc  simv*  vc_hdrs.h  ucli.key  urg* *.log  novas.* *.fsdb* &
#-----------------------------------
```

SIMULATION RESULTS ON VCS

SIMULATION RESULTS ON VCS

```
29457 Result is as Expected
29458 driver transaction is '{rst_n:'h1, alu_enable_a:'h1, alu_enable_b:'h0, alu_enable:'h1, alu_irq_clr:'h0, alu_op_a:AND_a, alu_op_b:XNOR_b, alu_in_a:'hff, alu_in_b:'hff, alu_irq:'h0,
      alu_out:'h0, alu_irq_last:'h0, alu_irq_clr_last:'h0} at time : 293655000
29459 monitor transaction is '{rst_n:'h1, alu_enable_a:'h1, alu_enable_b:'h0, alu_enable:'h1, alu_irq_clr:'h0, alu_op_a:AND_a, alu_op_b:XNOR_b, alu_in_a:'hff, alu_in_b:'hff, alu_irq:'h1,
      alu_out:'hff, alu_irq_last:'h0, alu_irq_clr_last:'h0} at time : 293655000
29460 time is 293655000 coverage of g1_tgl is 100.000000
29461 time is 293655000 coverage of g2 is 100.000000
29462 time is 293655000 coverage of g3 is 100.000000
29463 time is 293655000 coverage of g4 is 100.000000
29464 time is 293655000 coverage of g5 is 100.000000
29465 scoreboard transaction is '{rst_n:'h1, alu_enable_a:'h1, alu_enable_b:'h0, alu_enable:'h1, alu_irq_clr:'h0, alu_op_a:AND_a, alu_op_b:XNOR_b, alu_in_a:'hff, alu_in_b:'hff, alu_irq:'h1,
      alu_out:'hff, alu_irq_last:'h0, alu_irq_clr_last:'h0} at time : 293655000
29466 Result is as Expected
29467 $finish called from file "alu_tb.sv", line 761.
29468 $finish at simulation time            293655000
29469
29470 ------------------------------------------------------------------------
29471 VCS Coverage Metrics: during simulation line, cond, FSM, branch, tgl was monitored
29472 ------------------------------------------------------------------------
29473           V C S   S i m u l a t i o n   R e p o r t
29474 Time: 293655000 ps
29475 CPU Time:      1.440 seconds;       Data structure size:   0.0Mb
29476 Thu Dec  7 02:29:45 2023
```

SIMULATION RESULTS ON VCS



SIMULATION RESULTS ON VCS

Total Coverage Summary

SCORE	LINE	COND	TOGGLE	FSM	BRANCH	GROUP
100.00	100.00		100.00			100.00

Hierarchical coverage data for top-level instances

SCORE	LINE	COND	TOGGLE	FSM	BRANCH	NAME
100.00	100.00		100.00			top_tb

Total Module Definition Coverage Summary

SCORE	LINE	COND	TOGGLE	FSM	BRANCH
100.00	100.00		100.00		

Total Groups Coverage Summary

SCORE	WEIGHT
100.00	1

SIMULATION RESULTS ON VCS

Total groups in report: 5

NAME	SCORE	WEIGHT	GOAL	AT LEAST	PER INSTANCE	AUTO BIN MAX	PRINT MISSING	COMMENT
$unit::subscriber::g1_tgl	100.00	1	100	1	0	64	64	
$unit::subscriber::g2	100.00	1	100	1	0	64	64	
$unit::subscriber::g3	100.00	1	100	1	0	64	64	
$unit::subscriber::g4_illegal	100.00	1	100	1	0	64	64	
$unit::subscriber::g5_irq	100.00	1	100	1	0	64	64	

SIMULATION RESULTS ON VCS

Summary for Group $unit::subscriber::g5_irq

CATEGORY	EXPECTED	UNCOVERED	COVERED	PERCENT
Variables	15	0	15	100.00
Crosses	7	0	7	100.00

Variables for Group $unit::subscriber::g5_irq

VARIABLE	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT	AT LEAST	AUTO BIN MAX	COMMENT
op_aaa	4	0	4	100.00	100	1	1	0	
op_bbb	4	0	4	100.00	100	1	1	0	
ALU_OUT	7	0	7	100.00	100	1	1	0	

Crosses for Group $unit::subscriber::g5_irq

CROSS	EXPECTED	UNCOVERED	COVERED	PERCENT	GOAL	WEIGHT	AT LEAST	PRINT MISSING	COMMENT
irq_events_a_cross	3	0	3	100.00	100	1	1	0	
irq_event_b_cross	4	0	4	100.00	100	1	1	0	

COVERAGE PLAN ON VERDI

COVERAGE PLAN ON VERDI

```
plan "ALU Verification PLAn";


    feature Basics;
        owner = "ahmed\n";
        feature Alu_rstn;
            measure Group Measure_2;
                source = "group: $unit::subscriber::g2.t4.rst_n";
            endmeasure
        endfeature
    endfeature
    feature Advanced;
        owner = "youssef\n";
        feature alu_in_a;
            measure Group Measure_2;
                source = "group: $unit::subscriber::g1_tgl.al_in_a0", "group: $unit::subscriber::g1_tgl.al_in_a1", "group: $unit::subscriber::g1_tgl.al_in_a2", "group:
  $unit::subscriber::g1_tgl.al_in_a3", "group: $unit::subscriber::g1_tgl.al_in_a4", "group: $unit::subscriber::g1_tgl.al_in_a5", "group: $unit::subscriber::g1_tgl.al_in_a6", "group:
  $unit::subscriber::g1_tgl.al_in_a7";
            endmeasure
        endfeature
```

RESOURCES

RESOURCES

1. https://edaplayground.com/x/pgqd
2. https://drive.google.com/drive/folders/1J8qK3HRA0X3-30uCedGeiX85ineHsu5D?usp=sharing

THANK YOU