ASSIGNMENT 4 EXTRA

Q1

CODE :

Assignment_extra_4 > ALU_tb.sv

```
module ALU_4_bit_tb;
    reg clk;
    reg reset;
    reg [1:0] Opcode;          // The opcode
    reg signed [3:0] A;        // Input data A in 2's complement
    reg signed [3:0] B;        // Input data B in 2's complement
    wire signed [4:0] C;       // ALU output in 2's complement
    logic signed [4:0] expected;

    ALU_4_bit t (.*);
    bind ALU_4_bit ALU_sva sva (clk, reset, Opcode, A, B, C);

    // Clock generation
    initial begin
        clk = 0;
        forever #1 clk = ~clk;
    end

    initial begin

        repeat (10000) begin
            Opcode = $random;
            A = $random;
            B = $random;
            reset = $random;

            @(negedge clk);

        end

        $stop;
    end
endmodule
```

Assignment_extra_4 > ALU_sva.sv

```
module ALU_sva (
    input logic clk,
    input logic reset,
    input logic [1:0] Opcode,
    input logic signed [3:0] A,
    input logic signed [3:0] B,
    input logic signed [4:0] C
);

    // A + B
    property add_check;
        @(posedge clk) disable iff (reset)
            (Opcode == 2'b00) |=> (C == ($past(A) + $past(B)));
    endproperty
    assert property (add_check) else $error("Addition failed: C != A + B");
    cover property (add_check);

    // A - B
property sub_check;
    @(posedge clk) disable iff (reset)
        (Opcode == 2'b01) |=> (C == ($past(A) - $past(B)));
endproperty
    assert property (sub_check) else $error("Subtraction failed: C != A - B");
    cover property (sub_check);

    // ~A (bitwise NOT of A)
    property not_a_check;
        @(posedge clk) disable iff (reset)
            (Opcode == 2'b10) |=> (C == ~$past(A));
    endproperty
    assert property (not_a_check) else $error("Bitwise NOT of A failed");
    cover property (not_a_check);

    // Reduction OR of B
    property reduction_or_check;
        @(posedge clk) disable iff (reset)
            (Opcode == 2'b11) |=> (C == (|$past(B)));
    endproperty
    assert property (reduction_or_check) else $error("Reduction OR of B failed");
    cover property (reduction_or_check);

    // Reset output
    property reset_check;
        @(posedge clk) disable iff (!reset)
            (reset) |=> (C == 5'b0);
    endproperty
    assert property (reset_check) else $error("Reset failed: C != 0");
    cover property (reset_check);

endmodule
```

Assignment_extra_4 > ALU.do

```
vlib work
vlog ALU.v ALU_sva.sv ALU_tb.sv +cover -covercells
vsim -voptargs=+acc work.ALU_4_bit_tb -cover
add wave *
coverage save ALU_tb.ucdb -onexit
run -all
```

Cover Directives

Name	Language	Enabled	Log	Count	AtLeast	Limit	Weight	Cmplt %	Cmplt graph	Included	Memory	Peak Memory	Peak Memory Time	Cumulative Threa
/ALU_4_bit_tb/t/sv...	SVA	✓	Off	2454	1	Unli...	1	100%		✓	0	0	0 ns	
/ALU_4_bit_tb/t/sv...	SVA	✓	Off	671	1	Unli...	1	100%		✓	0	0	0 ns	
/ALU_4_bit_tb/t/sv...	SVA	✓	Off	650	1	Unli...	1	100%		✓	0	0	0 ns	
/ALU_4_bit_tb/t/sv...	SVA	✓	Off	613	1	Unli...	1	100%		✓	0	0	0 ns	
/ALU_4_bit_tb/t/sv...	SVA	✓	Off	685	1	Unli...	1	100%		✓	0	0	0 ns	

```
=================================================================================

Assertion Coverage:
    Assertions                        5         5         0   100.00%
---------------------------------------------------------------------------
Name                  File(Line)                    Failure      Pass
                                                    Count        Count
---------------------------------------------------------------------------
/ALU_4_bit_tb/t/sva/assert__reset_check
                      ALU_sva.sv(47)                     0          1
/ALU_4_bit_tb/t/sva/assert__reduction_or_check
                      ALU_sva.sv(39)                     0          1
/ALU_4_bit_tb/t/sva/assert__not_a_check
                      ALU_sva.sv(31)                     0          1
/ALU_4_bit_tb/t/sva/assert__sub_check
                      ALU_sva.sv(23)                     0          1
/ALU_4_bit_tb/t/sva/assert__add_check
                      ALU_sva.sv(15)                     0          1

Directive Coverage:
    Directives                        5         5         0   100.00%

DIRECTIVE COVERAGE:
```

```
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                         34        34         0   100.00%

================================Toggle Details================================

Toggle Coverage for instance /ALU_4_bit_tb/t/sva --

                                          Node      1H->0L      0L->1H  "Coverage"
                                          ----------------------------------------
                                        A[0-3]           1           1      100.00
                                        B[0-3]           1           1      100.00
                                        C[0-4]           1           1      100.00
                                     Opcode[0-1]         1           1      100.00
                                           clk           1           1      100.00
                                         reset           1           1      100.00

Total Node Count     =         17
Toggled Node Count   =         17
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (34 of 34 bins)
```

Q2

Assignment_extra_4 > assertion2.sv

```
module assertion2 ( input logic clk,request,grant,frame,irdy
);
    property prop1;
    @(posedge clk) $rose (request) |=> ##[2:5] grant;
    endproperty
    label1 : assert property (prop1);
    label2 : cover property (prop1);

    property prop2;
    @(posedge clk) $rose (grant) |-> ($fell(frame) && $fell(irdy));
    endproperty
    label3 : assert property (prop2);
    label4 : cover property (prop2);

    property prop3;
    @(posedge clk) ($rose (frame) && $rose (irdy)) |=> $fell(grant);
    endproperty
    label5 : assert property (prop3);
    label6 : cover property (prop3);

endmodule
```

Q3

Assignment_extra_4 > assertion3.sv

```
module assertion2 ( input logic [0:1] cs,
                    input logic clk,get_data
);
    parameter IDLE = 2'b00;
    parameter WAITO = 2'b01;
    parameter GEN_BLK_ADDR = 2'b11;

    property prop_cs;
    @(posedge clk) $onehot(cs);
    endproperty
    label1 : assert property (prop_cs);
    label2 : cover property (prop_cs);

    property prop;
    @(posedge clk) (cs == IDLE && get_data) |=> (cs == GEN_BLK_ADDR) ##64 (cs == WAITO);
    endproperty
    label3 : assert property (prop);
    label4 : cover property (prop);


endmodule
```