**Module: SV2**
**Section:** Direct Programming Interface **Task:1**
[EDA Playground Link]

The primary objective is to create a reference ALU model in C, import the function into the SystemVerilog testbench using DPI, and verify the functionality of the hardware ALU model against the C reference model by comparing their outputs.

## 2. C Function Implementation

The ALU reference model is implemented in C, where the function **`alu_reference_model`** performs the ALU operations based on input **A, B**, and the ALU selection (**`ALU_Sel`**). The C function computes the result and the carry-out signal, similar to the SystemVerilog ALU.

```
#include <svdpi.h>

// Author: Ahmed Raza
// Date: October 23, 2024
// Description: This function implements a reference ALU
model in C and is called from SystemVerilog using DPI.
// The function performs addition, subtraction,
multiplication, division, bitwise operations, shifts, and
comparisons.

void alu_reference_model(const svLogicVecVal* A,
                         const svLogicVecVal* B,
                         const svLogicVecVal* ALU_Sel,
                         svLogicVecVal* ALU_Out,
                         svLogic* Carry_Out) {

    unsigned char a = A[0].aval & 0xFF;    // 8-bit input A
    unsigned char b = B[0].aval & 0xFF;    // 8-bit input B
    unsigned char sel = ALU_Sel[0].aval & 0xF; // 4-bit ALU
selection
    unsigned char result = 0;             // Result
    unsigned short tmp = 0;               // Temporary for
carry calculation
```

```
    *Carry_Out = 0;                          // Initialize carry
out

    // ALU operations
    if (sel == 0x0) { // Addition
        tmp = a + b;
        result = tmp & 0xFF;
        *Carry_Out = (tmp > 0xFF); // Set carry if overflow
    }
    else if (sel == 0x1) { // Subtraction
        result = a - b;
        *Carry_Out = (a < b); // Set carry if borrow
    }
    else if (sel == 0x2) { // Multiplication
        result = a * b;
    }
    else if (sel == 0x3) { // Division
        result = (b != 0) ? (a / b) : 0; // Handle division
by zero
    }
    else if (sel == 0x4) { // Logical shift left
        result = a << 1;
    }
    else if (sel == 0x5) { // Logical shift right
        result = a >> 1;
    }
    else if (sel == 0x6) { // Rotate left
        result = (a << 1) | (a >> 7);
    }
    else if (sel == 0x7) { // Rotate right
        result = (a >> 1) | (a << 7);
    }
    else if (sel == 0x8) { // Logical AND
        result = a & b;
    }
```

```
    else if (sel == 0x9) { // Logical OR
        result = a | b;
    }
    else if (sel == 0xA) { // Logical XOR
        result = a ^ b;
    }
    else if (sel == 0xB) { // Logical NOR
        result = ~(a | b);
    }
    else if (sel == 0xC) { // Logical NAND
        result = ~(a & b);
    }
    else if (sel == 0xD) { // Logical XNOR
        result = ~(a ^ b);
    }
    else if (sel == 0xE) { // Greater comparison
        result = (a > b) ? 1 : 0;
    }
    else if (sel == 0xF) { // Equal comparison
        result = (a == b) ? 1 : 0;
    }

    // Set output
    ALU_Out[0].aval = result;
    ALU_Out[0].bval = 0;  // No X or Z states, so set bval to
0
}
```

## 3. SystemVerilog Testbench

The testbench invokes the ALU function using DPI and compares the expected values from the C function with the actual values from the SystemVerilog-based ALU.

**Updated Testbench: alu_tb.sv**

```
`timescale 1ns / 1ps

module tb_alu;

// Import Function for ALU using DPI
import "DPI-C" function void alu_reference_model(
    input logic [7:0] A,
    input logic [7:0] B,
    input logic [3:0] ALU_Sel,
    output logic [7:0] ALU_Out,
    output logic Carry_Out
);

// Inputs
logic [7:0] A, B;
logic [3:0] ALU_Sel;

// Outputs
logic [7:0] ALU_Out;
logic CarryOut;

// Expected Outputs variables
logic [7:0] exp_result;
logic exp_carry;

// Instantiate DUT
alu test_unit (
    .A(A),
    .B(B),
    .ALU_Sel(ALU_Sel),
```

```
    .ALU_Out(ALU_Out),
    .CarryOut(CarryOut)
);

integer i;

initial begin
    // Initial values
    A = 8'h0A;
    B = 8'h02;
    ALU_Sel = 4'h0;

    for (i = 0; i <= 15; i = i + 1) begin
        #1;

        // Call ALU function and store the result in expected
output variables
        alu_reference_model(A, B, ALU_Sel, exp_result,
exp_carry);

        // Compare expected and actual values
        if (exp_carry != CarryOut || exp_result != ALU_Out)
            $display("Test Failed: Expected = %h, Actual =
%h", exp_result, ALU_Out);
        else
            $display("Test Passed: Expected = %h, Actual =
%h", exp_result, ALU_Out);

        #10;
        ALU_Sel = ALU_Sel + 4'h1;
    end

    // Additional test cases
    A = 8'hF6;
    B = 8'h0A;
```

```
end

endmodule
```

## 4. Output

The output screenshots demonstrate the comparison between the expected and actual ALU values. The testbench iterates over all ALU operations, displaying "Test Passed" or "Test Failed" based on the results of the comparison.

Screenshot of Simulation Output:

```
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64;  Oct 24 03:00 2024
test Pass: Expected=234, Actual=234
test Pass: Expected= 10, Actual= 10
test Pass: Expected= 96, Actual= 96
test Pass: Expected=  1, Actual=  1
test Pass: Expected=244, Actual=244
test Pass: Expected=125, Actual=125
test Pass: Expected=245, Actual=245
test Pass: Expected=125, Actual=125
test Pass: Expected=240, Actual=240
test Pass: Expected=250, Actual=250
test Pass: Expected= 10, Actual= 10
test Pass: Expected=  5, Actual=  5
test Pass: Expected= 15, Actual= 15
test Pass: Expected=245, Actual=245
test Pass: Expected=  1, Actual=  1
test Pass: Expected=  0, Actual=  0
          V C S   S i m u l a t i o n   R e p o r t
Time: 276000 ps
CPU Time:      0.590 seconds;       Data structure size:   0.0Mb
Thu Oct 24 03:00:47 2024
Done
```

The comparison checks for every operation, such as addition, subtraction, multiplication, and various logical operations, ensuring that the results from the C reference model match the actual DUT outputs.