**Module: SV 2**

**Section:** Direct Programming Interface **Task:** DPI

**Direct Programming Interface -** [**EDA Link**](https://www.edaplayground.com/x/XmbB)

Task

* **C function:**

// Author: Noman Rafiq

// Dated: Oct 21, 2024

#include <svdpi.h>

void alu\_reference\_model(const svLogicVecVal\* A,

const svLogicVecVal\* B,

const svLogicVecVal\* ALU\_Sel,

svLogicVecVal\* ALU\_Out,

svLogic\* Carry\_Out) {

// Type-Cast values to 8-bit vectors

unsigned char a = (A->aval) & 0xFF;

unsigned char b = (B->aval) & 0xFF;

unsigned char select = (ALU\_Sel->aval) & 0xF; // Use only last 4-bits

unsigned char result;

// printf("a = %0d, b = %0d, select = %0d, A->aval = %0d, B->aval = %0d", a, b, select, A->aval, B->aval);

switch(select) {

// ADD CASE

case 0:

result = a + b;

\*Carry\_Out = (result < a)? 1:0;

// printf("A = %0d, B = %0d, ALU\_Out(ADD) = %0d\n", A, B, ALU\_Out);

// printf("C Side :: Select = %0d\n", select);

break;

// SUB CASE

case 1:

result = a - b;

\*Carry\_Out = (result > a)? 1:0;

break;

// MULT CASE

case 2:

result = a \* b;

\*Carry\_Out = 0;

break;

// DIV CASE

case 3:

result = (b != 0) ? a / b : 0;

\*Carry\_Out = 0;

break;

// Left-Shift CASE

case 4:

result = a << 1;

\*Carry\_Out = (a & 0x80)? 1:0;

break;

// Right-Shift CASE

case 5:

result = a >> 1;

\*Carry\_Out = (a & 0x01)? 1:0;

break;

// Rotate-Left CASE

case 6:

result = (a << 1) | (a >> 7);

\*Carry\_Out = 0;

break;

// Rotate-Right CASE

case 7:

result = (a >> 1) | (a << 7);

\*Carry\_Out = 0;

break;

// AND CASE

case 8:

result = a & b;

\*Carry\_Out = 0;

break;

// OR CASE

case 9:

result = a | b;

\*Carry\_Out = 0;

break;

// XOR CASE

case 10:

result = a ^ b;

\*Carry\_Out = 0;

break;

// NOR CASE

case 11:

result = ~(a | b);

\*Carry\_Out = 0;

break;

// NAND CASE

case 12:

result = ~(a & b);

\*Carry\_Out = 0;

break;

// XNOR CASE

case 13:

result = ~(a ^ b);

\*Carry\_Out = 0;

break;

// Greater CASE

case 14:

result = (a > b) ? 1:0;

\*Carry\_Out = 0;

break;

// Equal CASE

case 15:

result = (a == b) ? 1:0;

\*Carry\_Out = 0;

break;

default:

result = a + b;

\*Carry\_Out = (result < a)? 1:0;

break;

}

// Assign values to Actual variable

ALU\_Out->aval = result;

ALU\_Out->bval = 0;

// printf("result = %0d", result);

}

* **Updated Testbench:**

`timescale 1ns / 1ps

module tb\_alu;

//Import Function for ALU

import "DPI-C" context function void alu\_reference\_model(input logic [7:0] A, B,

input logic [3:0] ALU\_Sel,

output logic [7:0] ALU\_Out,

output logic CarryOut

);

//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;

// Verilog code for ALU

integer i;

alu test\_unit(

A,B, // ALU 8-bit Inputs

ALU\_Sel,// ALU Selection

ALU\_Out, // ALU 8-bit Output

CarryOut // Carry Out Flag

);

initial begin

// hold reset state for 100 ns.

A = 8'h0;

B = 8'h0;

ALU\_Sel = 4'h0;

#100;

A = 8'hFA;

B = 8'hF0;

ALU\_Sel = 4'h0;

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

begin

#1;

//Call ALU function here and store the result in expected output variables

alu\_reference\_model(A, B, ALU\_Sel, exp\_result, exp\_carry);

if (exp\_carry != CarryOut && exp\_result != ALU\_Out)

$display("test Fail: Expected=%d, Actual=%d", exp\_result, ALU\_Out);

else

$display("test Pass: Expected=%d, Actual=%d", exp\_result, ALU\_Out);

#10;

ALU\_Sel = ALU\_Sel + 4'h1;

end;

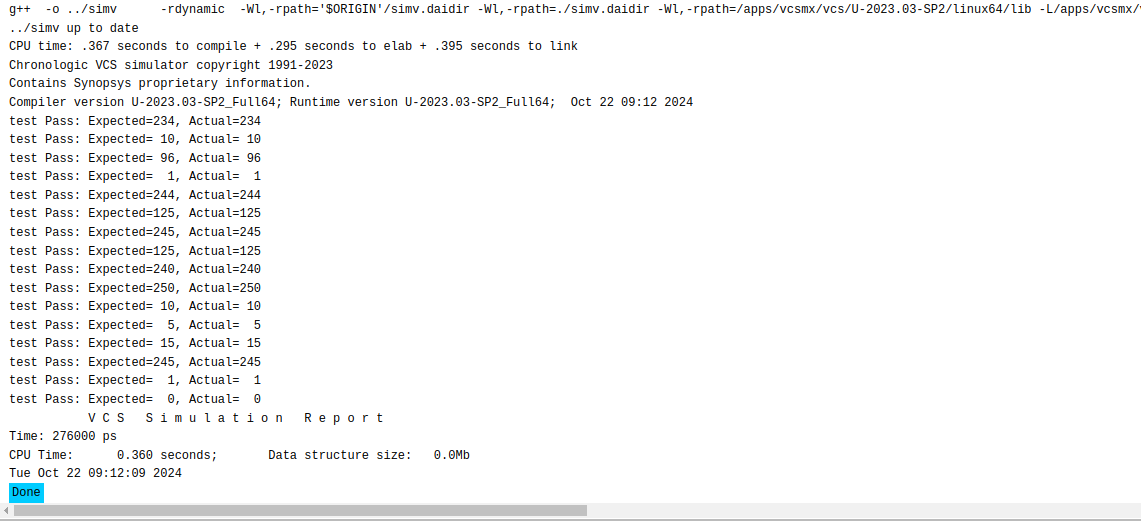
A = 8'hF6;

B = 8'h0A;

end

endmodule

* **Output Screenshot:**

****