CST456 – Lab 2 – Part 1

Simple Testbench with Data Types and Package

Objective

The objective of this lab is to implement a simple testbench to exhaustively test the operations and operands of a synchronous arithmetic and logic unit (ALU) using data types from a package.

Design Under Verification (DUV)

The ALU contains the following ports:

Port Name	Signal Type	Direction	Number of Bits	Data Type
clk	Clock	Input	1	logic
op_start	Control	Input	1	logic
operation	Control	Input	2	operation_t (enum)
operand_a	Data	Input	8	operand_t
operand_b	Data	Input	8	operand_t
result	Data	Output	16	result_t

At the rising edge of the clock, if op_start is asserted (== 1’b1) then the operation, operand_a, and operand_b signals are latched into the ALU. Exactly two clock cycles later the result of the operation is read from the result port. The ALU supports the following operations:

Name	Value	Description
ADD	2’b00	operand_a is added to operand_b. Only the lower nine bits of the result are used.
MULT	2’b01	operand_a is multiplied with operand_b. This is an unsigned operation and all sixteen bits of the result are used.
OR	2’b10	operand_a is bitwise ORed with operand_b. Only the lower eight bits of the result are used.
AND	2’b11	operand_a is bitwise ANDed with operand_b. Only the lower eight bits of the result are used.

Figure 1 shows a timing diagram for the addition operation, where 1 + 3 = 4.

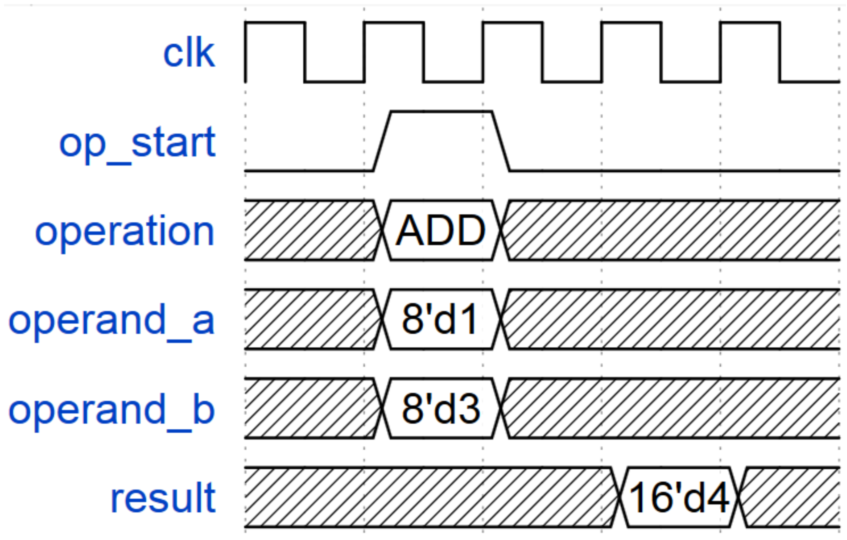

Figure 1 - Timing diagram for an addition operation.

Testbench

Edit the tb.sv file and create a testbench to exhaustively test all the operations and operand combinations that the ALU supports. The DUV uses data types from the typedef_pkg package declared in typedef_pkg.sv.

Create a do-while loop and two nested for-loops. The do-while loop enumerates through all four operations, the first for-loop enumerates through all 256 values for operand_a, and the second for-loop enumerates through all 256 values for operand_b. Use the `FAIL_UNLESS_EQUAL macro to check the expected result with the tested result. The DUV does not contain any bugs, so the testing is expected to be successful.

Simulate your testbench in Xilinx Vivado (**version 2023.2**) using following commands:

(simulation commands must be run under SIM directory)

```
call C:\Xilinx\Vivado\2023.2\bin\xvlog -nolog -sv  ../SRC/tb.sv  ../SRC/duv.sv

call C:\Xilinx\Vivado\2023.2\bin\xelab -debug typical -top tb -snapshot duv_tb_snapshot

call C:\Xilinx\Vivado\2023.2\bin\xsim   duv_tb_snapshot -R

call C:\Xilinx\Vivado\2023.2\bin\xsim   duv_tb_snapshot  --tclbatch  xsim_cfg.tcl

call C:\Xilinx\Vivado\2023.2\bin\xsim   --gui  duv_tb_snapshot.wdb
```

Lab Submission

Zip the entire contents of the lab directory and submit it to Canvas.