[Lab 1] Introduction to Verilog

[Lab 1] Introduction to Verilog

Seungho Lee (seungho.lee@snu.ac.kr)

Overview

In this project, you will learn how to describe digital circuits in Verilog and how to use design tools (e.g., Vivado Design Suite). Your assignment for next two weeks is to implement a simple 16-bit ALU, and it is the first step in starting the course project.

Goal

- Understand how to describe digital circuits in Verilog

- Implement an ALU (which is a basic functional unit in CPUs)

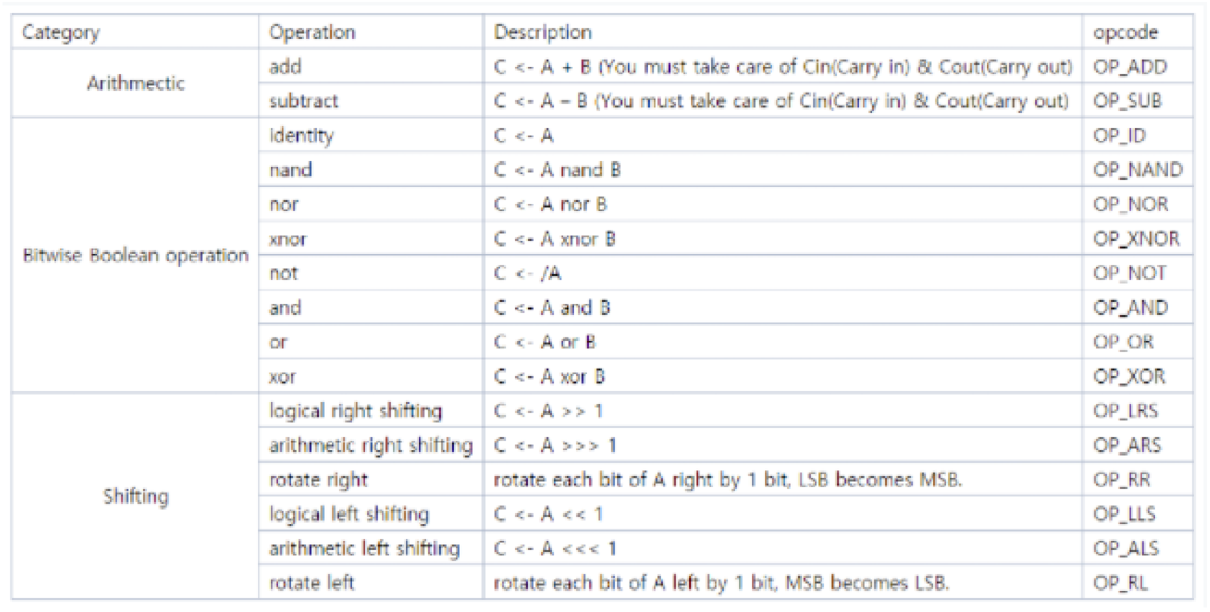
Verilog and Xilinx Vivado Desing Suite

Before starting your assignment, you should know how to describe digital circuits (e.g., ALU) in Verilog which is the most widely used and well-supported hardware description language (HDL). For students who are not familiar with Verilog programming, I posted the Verilog tutorial slides "**Introduction_to_Verilog.pdf"**. The slides include the basics of Verilog programming. You can find values, data types, operators in Verilog, and you can also find how to describe interfaces and behavioral models of your digital circuits. *If you have any question about Verilog and slides, please send an email to TAs or post questions on the etl Lab01 Q&A board.*

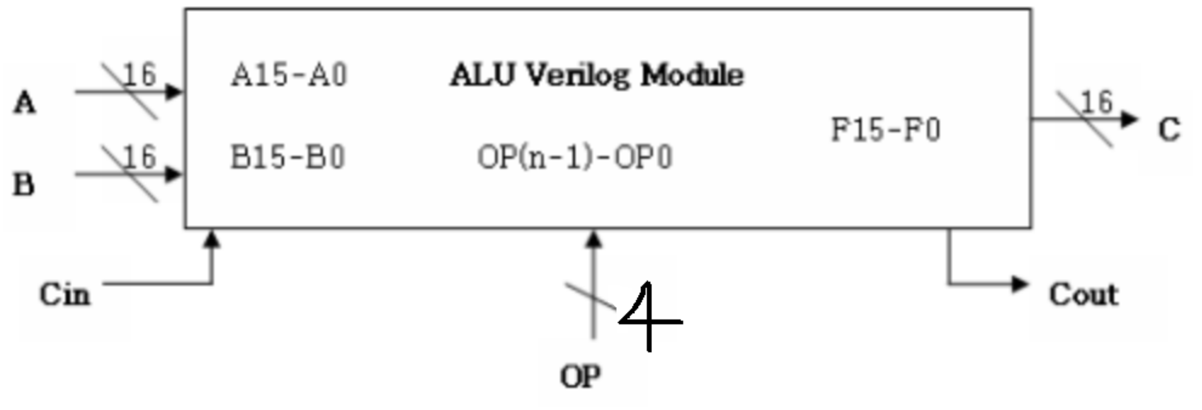
Second, to implement and verify your digital circuit, you will use Xilinx Vivado Design Suite. *Note that Xilinx Vivado Design Suite and its license are installed on computers in 301-207*. However, if you want to install Xilinx Vivado Design Suite on your computer, please follow instructions in the material **"Lab00_Vivado_supplement.pdf"**. The material also includes how to create a project, and how to simulate your design.

Assignment: ALU

Your assignment for next two weeks is to design and implement a 16-bit ALU (please refer to below table and block diagram). The table and block diagram show the ALU specification such as operations and input/output ports. You must follow the ALU specification and input/output ports.



Category	Operation	Description	opcode
Arithmectic	add	C <- A + B (You must take care of Cin(Carry in) & Cout(Carry out)	OP_ADD
	subtract	C <- A – B (You must take care of Cin(Carry in) & Cout(Carry out)	OP_SUB
Bitwise Boolean operation	identity	C <- A	OP_ID
	nand	C <- A nand B	OP_NAND
	nor	C <- A nor B	OP_NOR
	xnor	C <- A xnor B	OP_XNOR
	not	C <- /A	OP_NOT
	and	C <- A and B	OP_AND
	or	C <- A or B	OP_OR
	xor	C <- A xor B	OP_XOR
Shifting	logical right shifting	C <- A >> 1	OP_LRS
	arithmetic right shifting	C <- A >>> 1	OP_ARS
	rotate right	rotate each bit of A right by 1 bit, LSB becomes MSB.	OP_RR
	logical left shifting	C <- A << 1	OP_LLS
	arithmetic left shifting	C <- A <<< 1	OP_ALS
	rotate left	rotate each bit of A left by 1 bit, MSB becomes LSB.	OP_RL



Cin and Cout are only used for arithmetic operations. (remember Cin and Cout are 1 bit signals)

- add: C = A + B + Cin , Cout = 1 when overflow occurred (i.e. C[16] = 1)

- sub: C = A – (B + Cin), Cout = 1 when underflow at the largest bit occurred (i.e. A < B + Cin)

Cout = 0 for other operations (ex. Bitwise Boolean Operation, Shifting)

Grading

Design: ALU specification

Again, you have to follow the ALU specification (input/output ports, data width, operations). *Note that your ALU should be a combinational circuit. Please DO NOT include the clock or reset ports.*

Functionality: correct results on the testbench (ALU_TB.v)

A testbench is used to verify the functionality of designs. It will be very helpful for you to check the correctness of your modules. For example, a testbench instantiates your top-level module and drive input signals to the module. Then it checks output signals and compares them with expected values.

We posted the testbench file (ALT_TB.v) to help you to verify your ALU design. It contains many different test cases. Each test case drives different input values to fully verify the functionality. If your ALU passes all tests, you can see the message "Passed = 50, Failed = 0" on the console.

Note: you should modify definitions of opcodes to test your ALU design. Please refer to the "TODO" comments in the testbench file.

(Example)

#define OP_ADD 0

#define OP_SUB 1

#define OP_ID 2

Code review

I expect you to use **continuous assignment** or **blocking procedural assignment** for this assignment because it is common to use continuous and blocking assignments for combinational circuits.

I also expect you to make your code readable.

- Comments for important modules and lines

- Descriptive names for modules and variables

- Consistent code layout

Report

Please refer to **"Lab1_How_to_wirte_your_report.pdf"**

Submission

Due: Mon March 25 17:00 pm

>> There will be penalty for late submission. Specific late penalty will be noticed later.

Upload a single zip file including your (1) project folder, (2) report on the eTL board.

Teaching assistants

If you have any questions about this assignment, lectures, and homework, please use the Lab01 Q&A board.

Office: 1153-1 B301

TAs (arch19-ta@hpcs.snu.ac.kr)

Hunjun Lee (hunjunlee7515@snu.ac.kr)

Dongmoon Min (dongmoon.min@gmail.com)

Ilkwon Byun (1kwonByun@gmail.com)

Wonsik Lee (wonsik.lee@snu.ac.kr)

Heetaek Jeong (heetaek@snu.ac.kr)

Seungho Lee (seungho.lee@snu.ac.kr)