RAT Assignment #4
Arithmetic Logic Unit (ALU)

Learning Objectives:

- ✓ To understand the requirements of the Arithmetic Logic Unit (ALU) based on the RAT instructions it must support.
- ✓ To understand the design and operation of the ALU.

General Notes:

The ALU is part of the RAT's central processing unit (CPU). As with many computers, the ALU is responsible for performing operations beyond what can be labeled as "arithmetic" or "logic". In short, the ALU is responsible for performing the bit-crunching operations required by many of the RAT instructions. The ALU is a relatively complex device, but as you'll see in this experiment, this complexity can be successfully managed by the power of VHDL behavioral modeling.

Circuit Details

- There are a few examples of ALUs modeling using VHDL in the textbook. These will be a good starting point for this assignment.
- Try to keep your code as generic as possible.
- The ALU is a critical component for meeting timing requirements of the RAT MCU. Ensure no latches are present in the design.
- The ALU cannot contain long chains of gates. The elaborated design should be easy to trace and understand behaviorally.
- Not all instructions handle the C and Z flags in a similar manner. Make sure to reference the RAT Assembler Manual.
- The ALU can generate any Z and C value, but it doesn't have to save those values in the C and Z flag registers (the C and Z flags are implemented as flip-flops and are part of a future assignment).

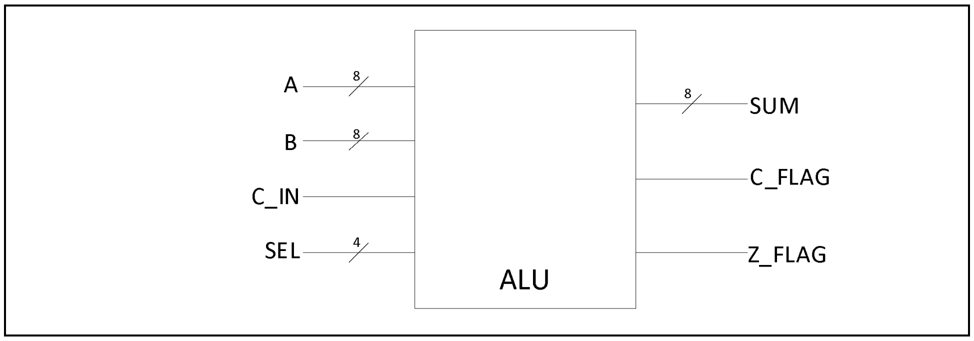

Figure 1: Black box diagram for the RAT ALU.

Assignment

Implement the 15 RAT arithmetic and logic functions in an ALU. Recall that the ALU is typically a combinatorial device. Refer to the RAT Assembler manual for an expanded explanation of each of the associated RAT instruction. Figure 1 shows the black box diagram for the ALU; Table 1 shows the SEL values you should use for the RAT instructions that require the ALU.

SEL	Instruction		SEL	Instruction
0000	ADD		1000	TEST
0001	ADDC		1001	LSL
0010	SUB		1010	LSR
0011	SUBC		1011	ROL
0100	CMP		1100	ROR
0101	AND		1101	ASR
0110	OR		1110	MOV
0111	EXOR		1111	unused

Table 1: ALU Selection table

Verification

Complete the test cases in Table 2 on the next page and include it with your RAT assignment report. Note that the ALU always has a "result" for all operations, as it is simply an output. If the C or Z flags aren't affected by the instruction, place an "X" in the respective column. Consider teach line in Table 2 to be independent of previous lines.

Implement these test cases in an automated testbench in order to verify the proper operation of your ALU. Ensure that your hand calculated results from Table 2 match the results from the simulator output. This means you must include all the operation in Table 2 in your testbench. Make sure you test the operations in the simulation in the same order they appear in Table 2.

Function	SEL	Arguments (A, B, Cin)	Expected SUM	Expected C_FLAG	Expected Z_FLAG	Time tested in ISim
ADD		0xAA, 0xAA, 0				
		0x0A, 0xA0, 1				
		0xFF, 0x01, 0				
ADDC		0xC8, 0x36, 1				
		0xC8, 0x37, 1				
SUB		0xC8, 0x64, 0				
		0xC8, 0x64, 1				
		0x64, 0xC8, 0				
SUBC		0xC8, 0x64, 0				
		0xC8, 0x64, 1				
		0x64, 0xC8, 0				
		0x64, 0xC8, 1				
CMP		0xAA, 0xFF, 0				
		0xFF, 0xAA, 0				
		0xAA, 0xAA, 0				
AND		0xAA, 0xAA, 0				
		0x03, 0xAA, 0				
OR		0xAA, 0xAA, 0				
		0x03, 0xAA, 0				
EXOR		0xAA, 0xAA, 0				
		0x03, 0xAA, 0				
TEST		0xAA, 0xAA, 0				
		0x55, 0xAA, 0				
LSL		0x01, 0x12, 0				
LSR		0x80, 0x33, 0				
		0x80, 0x43, 1				
ROL		0x01, 0xAB, 1				
		0xAA, 0xF2, 0				
ROR		0x80, 0x3C, 0				
		0x80, 0x98, 1				
ASR		0x80, 0x81, 0				
		0x40, 0xB2, 0				
MOV		0x50, 0x30, 0				
		0x43, 0x22, 1				

Table 2: ALU Test Cases for Verification