|  |  |
| --- | --- |
| NAME | ID STUDENT |
| Nguyễn Thị Hồng Nhung | 2051163 |
| Nguyễn Hoài Hiếu Ngân | 2051154 |
| Nguyễn Phạm Minh Khôi | 2151105 |

# OBJECTIVES

* Getting to know how to describe a floating-point arithmetic, which could do add, subtract, and multiply operations.
* Design and implement digital circuits using FSM.
* Download the circuit into the FPGA chip and test its functionality.

# PREPARATION FOR LAB 3

* Finish Pre Lab 3 at home.
* Students have to simulate all the exercises in Pre Lab 3 at home. All results (codes, waveform, RTL viewer, … ) have to be captured and submitted to instructors prior to the lab session.

*If not, students will not participate in the lab and be considered absent this session.*

# REFERENCE

Floating point numbers allow computers to perform operations on a wide range of numbers.

According to the IEEE standards, floating point numbers are of the form

**(-1)^S \* (1+F) \*2^E**

S is the sign bit, which determines whether the number is positive or negative. F – fraction – holds the significant bits of the floating point number.

E is the exponent that (1+F) is raised to.

A 32 bit floating point number is standard, however fo simpification, we will be using an 8 bit representation. We will have a sign bit, 3 bits for the exponent, and the remaining 4 bits will be devoted to the fraction. This will allow us to represent a resolution as small as 1/128 and the smallest number we can represent is 1/8. Eight-bit floating point numbers are not useful for performing extremely accurate calculations, but it does demonstrate the operation of a floating point adder.

# EXPERIMENT

***Objective:*** Implement a floating point adder/subtractor using System Verilog description and download the cicuit into the FPGA chip..

## Requirement:

You are desired to design a floating-point adder/subtractor should perform correctly in the normal cases. Besides several pins needs to indicate some extreme cases:

- Zero detection: when the result is zero, zero detection pin will be 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Operand 1 | A[7:0] | 8-bit normalized input |
| Operand 2 | B[7:0] | 8-bit normalized input |
| Selection | S | 1-bit input, addition/subtraction selection |
| Outputs | Operation result | Result[7:0] | 8-bit normalized output |
| Zero detection | Z | 1-bit ouput |

*Table 1: IO definition*

## Instruction:

Input A and B have 8 bits, in which the sign is represented by bit [7], the exponent value is represented by bit [6:4], and the remaining is for Fraction value. Output Result also need to be normalized as input signal.

The design have some sub-modules that perform floating point calculations:

* Identify which number is larger, which number is smaller.
* Indentify the amount to right shift the operand which has smaller exponent.
* Right shift fraction value of the smaller operand to align decimal points.
* Calculate the two's compliment of the shifted fraction, only needed in the case of subtraction or equivalent case (i.e. adding a negative number to a positive number).
* Add the two fractions together.
* Normalize the fraction and exponent value so it's back in floating point representation.
* Determine sign of the final value.
* Detect zero: the result is zero if the signs of A and B are different and there is no difference in the fraction and exponent.
  1. Create a new Quartus project for your circuit.
  2. Use switches as inputs, LEDRs as outputs.
  3. Compile your project. Download the circuit into the FPGA chip and test its functionality.

***Check:*** Your report has to show two results:

* The waveform to prove the circuit works correctly.
* The result of RTL viewer.





