Amirkabir University of Technology
(Tehran Polytechnic)

Computer Architecture
HW2 - FPU Design
Instructor: Dr. Mohammadreza Pourfard
TA: Javad Sobhani • Fall 2025

1. Project Introduction

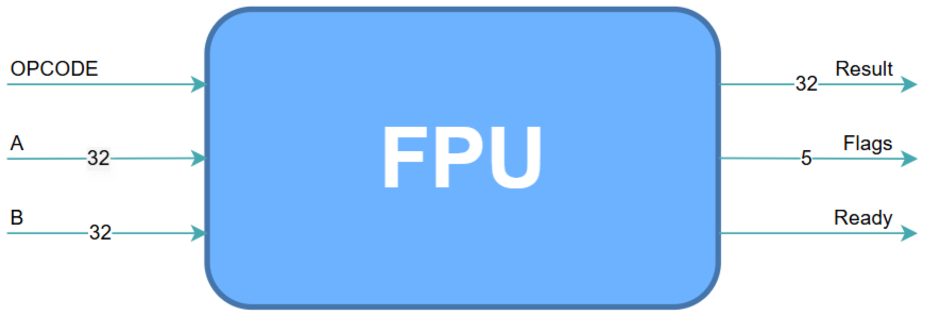
In this project, we aim to design a simple Floating Point Unit (FPU) based on the RISC-V architecture. The module will be programmed in VHDL and simulated using ModelSim.

We will first discuss the module and its parameters to further clarify the design goals you'll design for. Then the structure of the design is discussed, and a basic overview of the hardware is presented. Finally, the test condition and simulation steps are inspected.

2. FPU Module

2.1 Module Description

To be able to design the system, we first need to get familiar with the architecture of the module. The figure below shows the block diagram of the FPU block.



According to the figure above, the system has three major inputs.

- OPCODE: In this case, the opcode is simply one or zero
- Input A: 32-bit input as the first operand
- Input B: 32-bit input for the second operand

Given the opcode, addition or subtraction can be performed. Once the operation is finished, the result, along with the flags, is output, and a Ready flag goes high for one clock cycle.

The table below shows the operation of the system given the inputs.

OPCODE	Operand1	Operand2	Operation
0	A	B	$A + B$
1	A	B	$A - B$

2.2 Flags

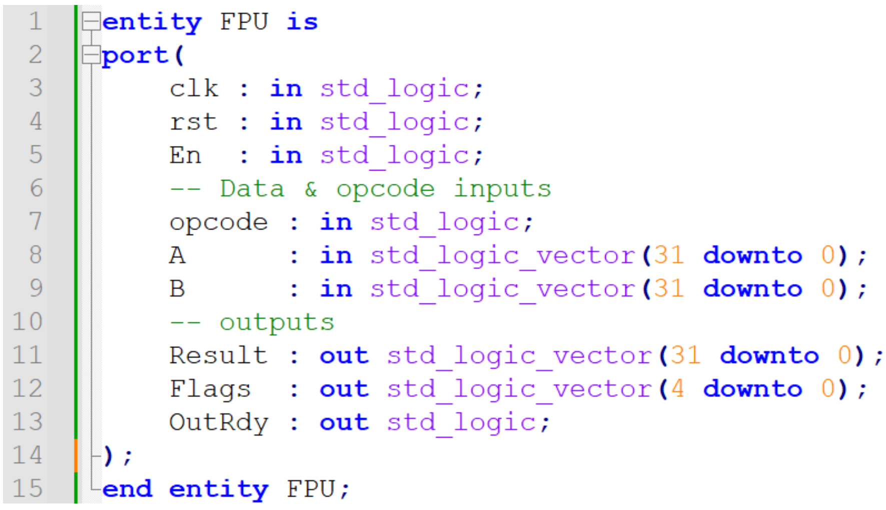
In case of some occurrences like overflow, underflow, or zero result, certain bits in the flag are set to indicate the system of these occurrences. The table below shows the details of the Flag design:

Bit	Flag	Symbol	Description	Example
0	Inexact	I	Result was rounded	1e20 + 3.14
1	Underflow	X	Result too small to represent	0x00000001 $-$ 0
2	Division by Zero	Z	Finite number ÷ 0	5.0 ÷ 0.0
3	Overflow	V	Result too large to represent	1e38 × 1e38
4	Invalid Operation	N	Invalid operation performed	0x7FC00000 $+$ 0

It's worth noting that the above flags are known as Exception Flags, and there is a separate set of flags called status flags, which will not be implemented in this design.
Furthermore, from the above table, The Division by zero is not utilized here but was mentioned due to design consistency. So in the Code will be left blank(0) and must never be set.

2.3 VHDL Module

Below is the definition of the FPU module in VHDL.



```
entity FPU is
port(
        clk : in std_logic;
        rst : in std_logic;
        En  : in std_logic;
        -- Data & opcode inputs
        opcode : in std_logic;
        A      : in std_logic_vector(31 downto 0);
        B      : in std_logic_vector(31 downto 0);
        -- outputs
        Result : out std_logic_vector(31 downto 0);
        Flags  : out std_logic_vector(4 downto 0);
        OutRdy : out std_logic;
);
end entity FPU;
```

In the previous diagram, we didn't show the clock, reset, and enable signals, but they are essential for the correct operation of the device.
The reset signal will only reset(0) the Result", "Flags", and .ºutRdy", and there is no need to reset the entire internal registers.

Furthermore, to perform any operation, the En pin needs to be set(1) along with other inputs.

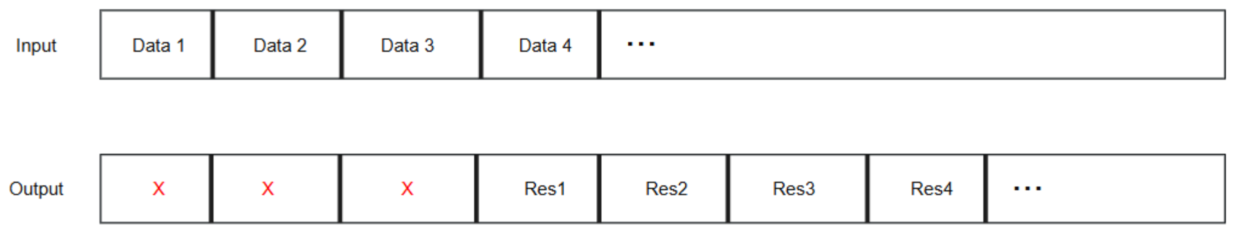
2.4 Flow of Operation

To use the module to perform either addition or subtraction, Below steps need to be taken: (assuming the clock signal is present)

1. The data and Enable pin need to be set to begin the operation.
2. once the operation is calculated, the result is output, and the OutRdy pin will go high.(Notice: The OutRdy pin needs to stay low while the valid data is not present.)

It's worth mentioning that the implementation needs to be handled in a pipelined manner. meaning that the input data can be cascaded one after another into the module, and after a certain delay, the valid data will be present at the output one after another, and the OutRdy pin will stay high as long as valid data is present.

Below is an example of a system with a 3-clock-cycle delay, but implemented in a pipelined manner.



2.5 Test & Simulations

The system needs to be tested under various conditions to check the validity of the module. The following tests are required to be performed:

- Overflow test
- Underflow test
- rounded number
- invalid operation attempt
- normal addition and subtraction
- pipeline test.

For each of the above categories, one example will suffice.

3. How To Submit

- The submission needs to be uploaded under HW2_StudentNum.
- Keep the structure of the directory given in the RAR file.
- attach a Report file along with the code as well.
- the Report file needs to be typed. Writing in LaTeX will have additional points.
- Designing a test sample in MATLAB will have additional points.
- Explaining the TCL file and its workings will have additional points.
- Clean and commented code is much appreciated.

DeadLine: December the 1st, 2025