Task 6

Compilation

In order to compile the alu and all test cases to bin folder this command must be run

make all

Usage

Exceute each program inside bin folder to run test cases of each module

Module overview

The program module dependencies are organized this way

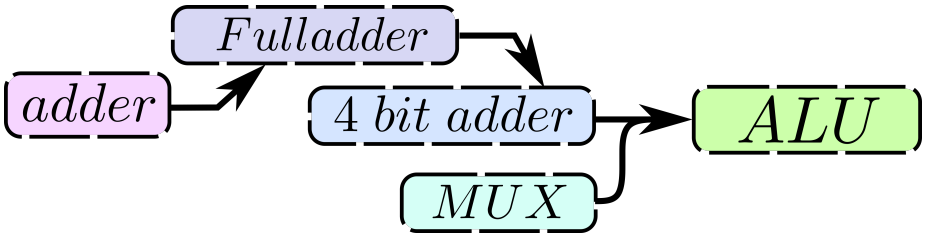

Figure 1: Module dependencies

Each module has its own executable file to run their tests. The modules are

- Mux: It connects the corresponding operation wire with corresponding output. It is an 8 input mux with inputs of size 4.
- $4\ bit\ adder$: Its function is to add 4 bit numbers and compute carry and overflow flags of that operations
- $Fulladder$: Adds 1 bit numbers with carry in, and carry out functionality
- $Adder$: Basic one bit number adder with carry functionality

Currently sum and substract are built from scratch, other operations use verilog built-in fucntions, altough, it would not cause any compatibility problem to have these functions manually coded.

Now we will analize in more depth how the program works

ALU overview

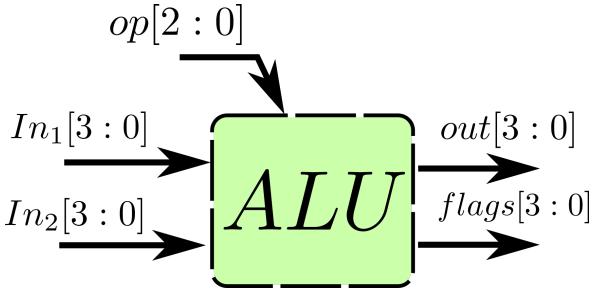

Figure 2: Alu in-out

The ALU makes sure the output and flags are correct accoring to opcodes and inputs.

Operations

There are 8 operations

- **addition** (000) : Implemented using 4 bit full adder
- **difference** (001): Implemented using 4 bit full adder using carry in and built-in not operator
- **and** (010): Implemented with built-in function
- **or** (011): Implemented with built-in function
- **not** (100): Implemented with built-in function
- **xor** (101): Implemented with built-in function
- **2'th complement** (110): Implemented with built-in function
- **shift left** (111): Implemented with built-in function

All 8 operations are connected into a wire array, then Mux comes into action and select according to opcode which action write to output Also another important detail is that operations that use only one number use input 1, and ignore input 2

There are four flags

- **carry/overflow** (Managed by 4 bit adder modules)
- **zero/negative** (Easily managed directly by ALU)

It is important to note that operations that do not use carry and overflow make these bits to be 0.

Mux

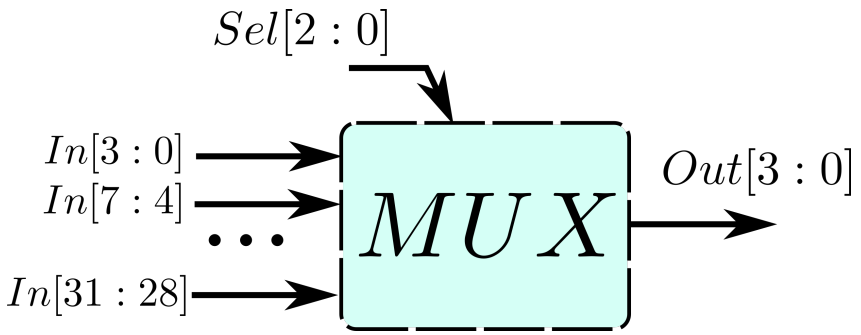

Figure 3: Mux in-out

This is a very simple module, implemented using vectorized expresions. It justs decides which input is wired to output according to selector input

Full 4bit Adder

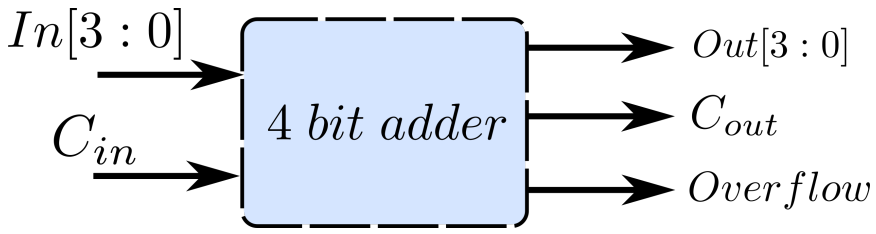

Figure 4: Full 4bit Adder in-out

This module adds two binary 4 bit number with carry-in, carry-out functionality. Also it computes the overflow flag by making the xor of the 'last bit' carry and the carry-out Also , to function this module uses 4 single full adders connected in cascade. We won't describe these modules in this file because they are standard and the sources are enough.