Victor Oliveira
Lightelligence Coding Challenge
Sunday, 10/02/2022

Design Questions:

1.It's hard to comment on PPA and timing since the division block is quite vague. I know that in my last company, we had a complex algorithm that allowed division to be done in 32 clock cycles, and the chip would be stalled during this time. Therefore, depending on the division block, there could be a lot of unexpected area being taken up, and it could affect performance if it takes much longer than other operations. Timing is hard to comment on because I'm not sure if it's possible to fit division into a single clock cycle, even with a slow clock speed. I don't think I used too many flops/gates, so the power usage should not be too high.

2. I used a case statement to implement the MUX used to select the correct output. This makes it very easy to add new functional units, one simply needs to add them to the enum within alu_pkg.svh and add the correct case to alu.sv. As for other data types, for signed integers it depends on how the functional blocks have been designed, I believe addition and subtractions should behave properly if 2's complement is used. Otherwise the functional blocks would require an input informing them whether the data is signed or unsigned. For inputs with fewer bits this ALU should behave as intended, for inputs with higher bits it would need to be modified.

3. Since this is only an 8 bit ALU, it should be fairly simple to verify that all combinations of inputs yield the expected results. It would take 256*256 combinations for each operation, for a total of 327680 combinations, which should not take much time since the operations are simple.

4. I think my solution is easy to read, and my comments should clarify any small questions. The code feels pretty straight forward since it's mostly just a MUX selecting what unit's output is the correct one, and whether it's valid or not, so I believe it should be easy to read.

5. I'm not too experienced with post-silicon, but I believe it would be helpful if one could design a small test board that allows one to control the inputs of the ALU and read its output. Depending on the ALU IC packaging, this could easily be done even on a breadboard, otherwise I would need to design an adapter that connects to the inputs/output of the ALU.

Notes:

- Typo in alu_pkg.svh line 30:
    - logic [NUM_WORTH_WIDTH-1:0] num_words;
    - should be logic [NUM_WORD_WIDTH-1:0] num_words;
- No carry? Mul output only 8 bits? Division outputs remainder but is unused?
- No accum register for MAC, will add one, assuming synchronous reset to 0
- Assuming reset_n signal is active high
- Assuming MAC operation outputs new value of accum register
- Should I assume all operations take a single cycle? I'm certain the division would take longer, but that would complicate things a lot
- I'm a little lost on the num_words parameter for alu_cmd_t
    - Is the goal to process one word at a time?
    - That's easy if every operation takes one clock cycle.
    - Otherwise, it would require each operation module to output a done or stall signal as well so that the system knows when to move to the next word