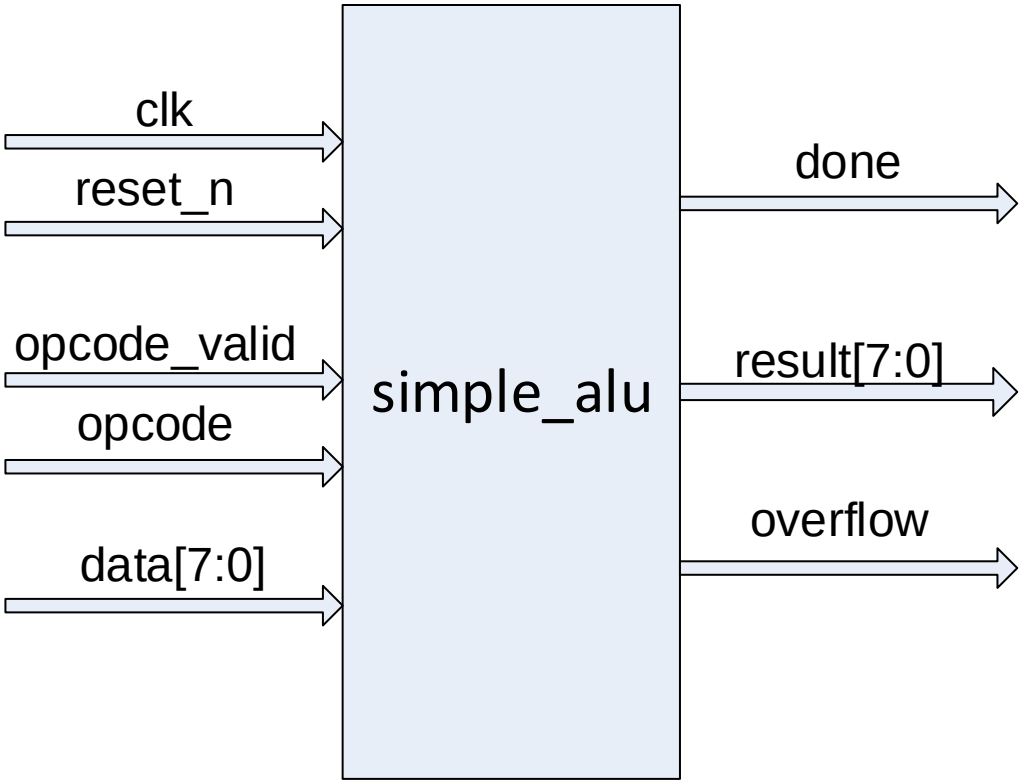
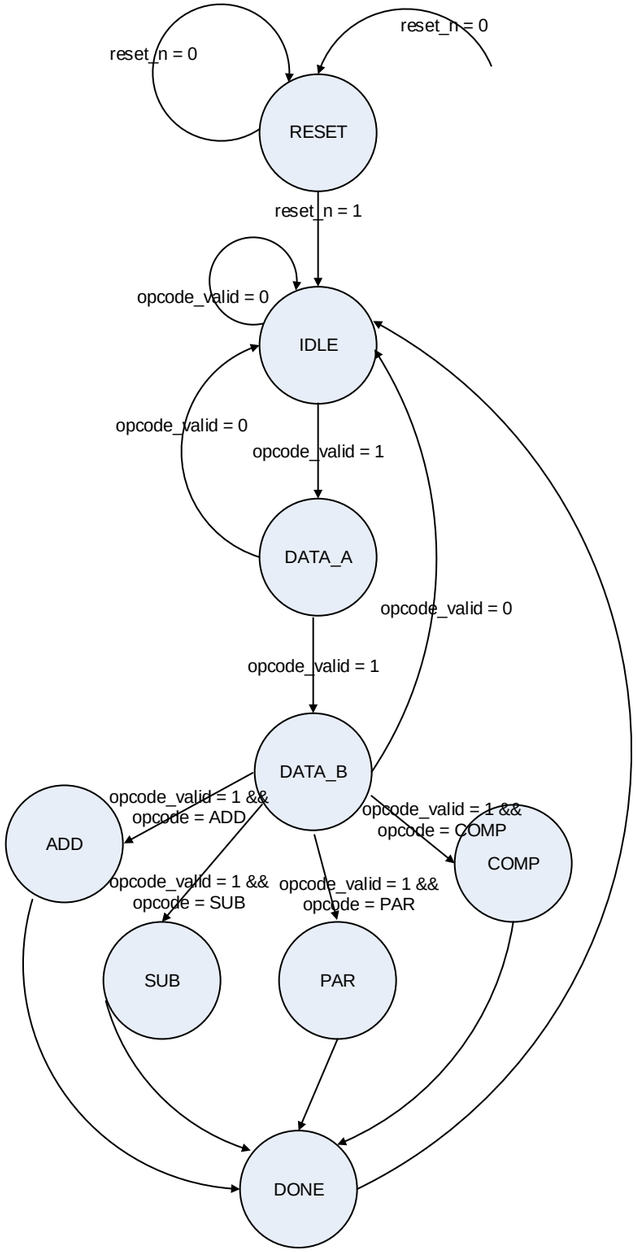


This is a simple ALU which supports 4 operations. Binary addition (ADD), Binary Subtraction (SUB), Bitwise parity calculation (PAR) and Bitwise comparator (COMP)

Opcode definition:
ADD: 2'b00
SUB: 2'b01
PAR: 2'b10
COMP: 2'b11

Description of inputs:
clk: Free running clock whose frequency can be set by a parameter (CLOCK_PERIOD). Default: 10ns
reset_n: Active low reset.
opcode_valid: active high input to control the internal FSM.
opcode: 1 bit opcode field that must be time-multiplexed for two bits needed internally for 4 opcodes.
Data[7:0]: 8-bit data field that also needs to be time-multiplexed for two operands.



Simple ALU

Note; ADD, SUB, PAR, COMP operations need two data operands. But as you can see simple_alu module has only one data input. That means data pins must be time multiplexed. First operand must be latched in DATA_A state and second operand must be latched in DATA_B state.
Similarly, there are four valid opcodes which needs a two bit field. However, the opcode input is only single bit. Bit[0] of opcode field must be latched in DATA_A state and bit[1] must be latched in DATA_B state.

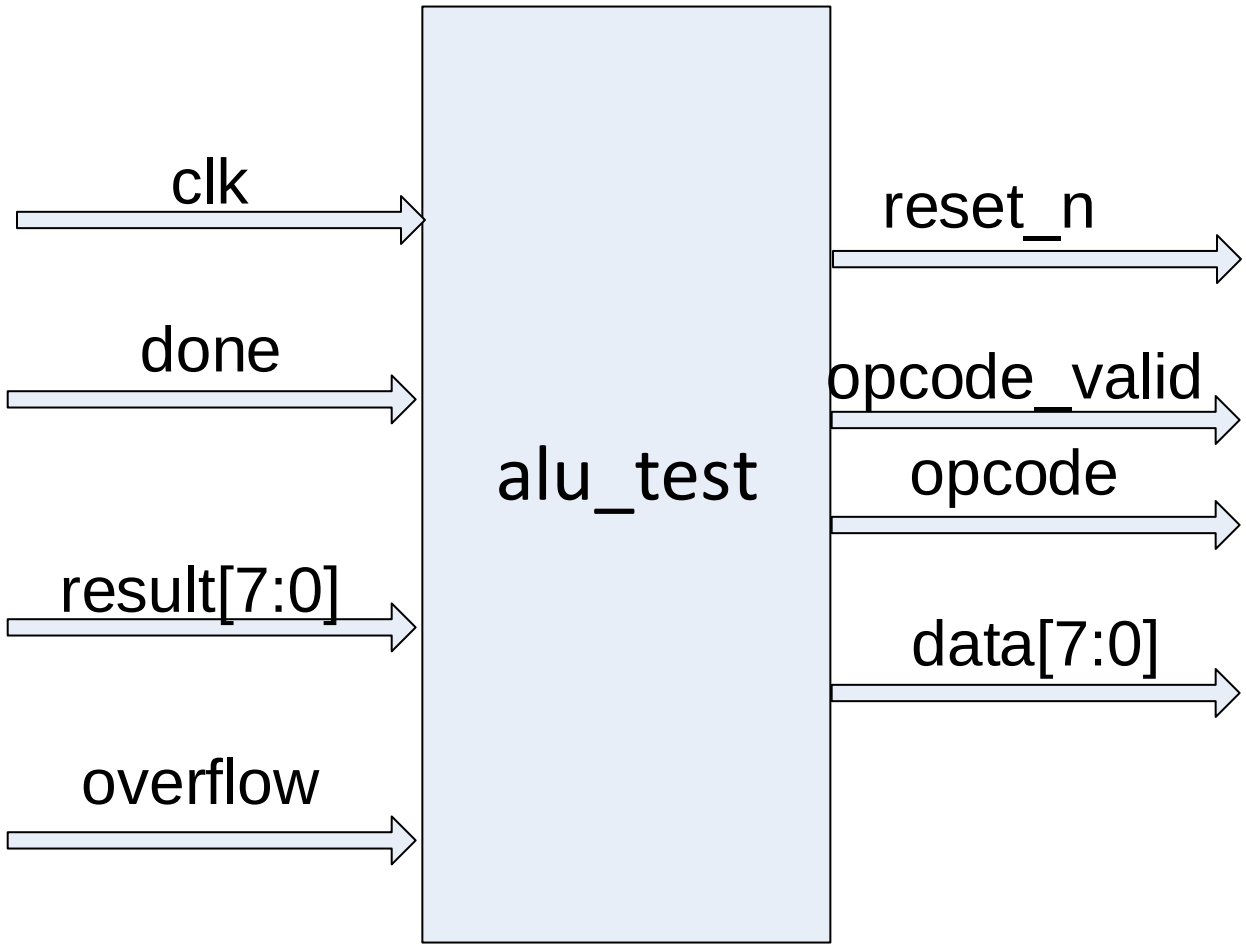
Test stimulus module (alu_test.v) will drive two operands and two bit of opcode separately on different clock cycles.

You must use gate-level logics (OR/NOR/AND/NAND/XOR/XNOR) for all of your low level functions. For example, you can create a module for 1bit full adder and then create a generic full adder with parameterized width. You need to use the generic full adder to get the result for ADD opcode. You can NOT simply say “result = data_a+data_b”, even though verilog language supports that.

The full adder must have two inputs and carry-in. The full subtractor must also have two inputs and borrow-in.

Truth table for Parity (PAR) function
A | B | Parity
---
0 | 0 | 0
0 | 1 | 1
1 | 0 | 1
1 | 1 | 0

Truth table for Comparator (COMP) function
A | B | Parity
---
0 | 0 | 1
0 | 1 | 0
1 | 0 | 0
1 | 1 | 1


clk
done
result[7:0]
overflow
alu_test
reset_n
opcode_valid
opcode
data[7:0]