Lab 5

MyHDL ALU

- Internal Signals
  - notb
  - mux1_out
  - and_out
  - or_out
  - adder_sum

```
# The function is called if any Signal on the sentitivity list has a new value
@always_comb
def alu1_logic():
    notb = not b
    # notb is an internal signal. we do not need define a MyHDL Signal
    mux1_out = notb if binvert else b
    # you can check the types of notb and b
    # print(type(notb), type(b))
    or_out = mux1_out or a
    and_out = mux1_out and a

    # 1 bit Adder
    a1 = a ^ mux1_out
    a2 = a1 ^ carryin
    b1 = a and mux1_out
    b2 = a1 and carryin
    addr_sum = a2
    carryout.next = b1 or b2

    #Opperation
    if operation == 0:
        result.next = and_out
    if operation == 1:
        result.next = or_out
    if operation == 2:
        result.next = addr_sum
    if operation == 3:
        result.next = 0
```

- This is the set of outputs, first mux1_out is calculated as notb if binvert and b if not binvert

- Next or_out and and_out are calculated with the return value mux1_out
- The adder is then used to calculate what the sum and new carry value are
- The opperation then gives the propper return value for the opperation being run
- 0 output
    -
```
(venv) ➜  ~/Documents/UCONN/Spring 2022/CSE3666/Labs/Lab5 python3 alu1.py 0
op a b cin bneg | cout res
00 0 0 0   0    | 0    0
00 1 0 0   0    | 0    0
00 0 1 0   0    | 0    0
00 1 1 0   0    | 1    1
00 0 0 1   0    | 0    0
00 1 0 1   0    | 1    0
00 0 1 1   0    | 1    0
00 1 1 1   0    | 1    1
00 0 0 0   1    | 0    0
00 1 0 0   1    | 1    1
00 0 1 0   1    | 0    0
00 1 1 0   1    | 0    0
00 0 0 1   1    | 1    0
00 1 0 1   1    | 1    1
00 0 1 1   1    | 0    0
00 1 1 1   1    | 1    0
```
- 1 output
    -
```
(venv) ➜  ~/Documents/UCONN/Spring 2022/CSE3666/Labs/Lab5 python3 alu1.py 1
op a b cin bneg | cout res
01 0 0 0   0    | 0    0
01 1 0 0   0    | 0    1
01 0 1 0   0    | 0    1
01 1 1 0   0    | 1    1
01 0 0 1   0    | 0    0
01 1 0 1   0    | 1    1
01 0 1 1   0    | 1    1
01 1 1 1   0    | 1    1
01 0 0 0   1    | 0    1
01 1 0 0   1    | 1    1
01 0 1 0   1    | 0    0
01 1 1 0   1    | 0    1
01 0 0 1   1    | 1    1
01 1 0 1   1    | 1    1
01 0 1 1   1    | 0    0
01 1 1 1   1    | 1    1
```
- 2 output

```
(venv) ➜  ~/Documents/UCONN/Spring 2022/CSE3666/Labs/Lab5 python3 alu1.py 2
op a b cin bneg | cout res
10 0 0 0   0    | 0    0
10 1 0 0   0    | 0    1
10 0 1 0   0    | 0    1
10 1 1 0   0    | 1    0
10 0 0 1   0    | 0    1
10 1 0 1   0    | 1    0
10 0 1 1   0    | 1    0
10 1 1 1   0    | 1    1
10 0 0 0   1    | 0    1
10 1 0 0   1    | 1    0
10 0 1 0   1    | 0    0
10 1 1 0   1    | 0    1
10 0 0 1   1    | 1    0
10 1 0 1   1    | 1    1
10 0 1 1   1    | 0    1
10 1 1 1   1    | 1    0
```

- 3 ouput

```
(venv) ➜  ~/Documents/UCONN/Spring 2022/CSE3666/Labs/Lab5 python3 alu1.py 3
op a b cin bneg | cout res
11 0 0 0   0    | 0    0
11 1 0 0   0    | 0    0
11 0 1 0   0    | 0    0
11 1 1 0   0    | 1    0
11 0 0 1   0    | 0    0
11 1 0 1   0    | 1    0
11 0 1 1   0    | 1    0
11 1 1 1   0    | 1    0
11 0 0 0   1    | 0    0
11 1 0 0   1    | 1    0
11 0 1 0   1    | 0    0
11 1 1 0   1    | 0    0
11 0 0 1   1    | 1    0
11 1 0 1   1    | 1    0
11 0 1 1   1    | 0    0
11 1 1 1   1    | 1    0
```

- All outputs were verified to be correct, I know that these are also correct because the values given by the adder are done out directly, and modeled. The inputs are all in the correct positions after they are returned.