Group 24

Gorantla Thoyajakshi – 21CS10026

Ashwin Prasanth – 21CS30009

Computer Organization and Architecture Lab

VERILOG Assignment 3, Question 1

- We made eight different modules for all the required operations in the ALU.

Operator modules:

- To implement **adder** module, we created a 4bit carry look ahead adder and cascaded two such units to get the sum of the 8-bit inputs.
- To implement **subtractor** module, we initially complemented the second input and then added it to the first input using the adder module.
- To implement **move** module, we took an 8-bit input and assigned it to result.
- To implement **leftshift** module, we took an 8-bit input and assigned the result to the input left-shifted by 1.
- To implement **rightshift** module, we took an 8-bit input and assigned the result to the input right-shifted by 1.
- To implement **and** module, we took two 8-bit inputs and assigned the result to the bitwise and of the inputs.
- To implement **or** module, we took two 8-bit inputs and assigned the result to the bitwise or of the inputs.
- To implement **not** module, we took an 8-bit input and assigned the result to the bitwise not of the input.

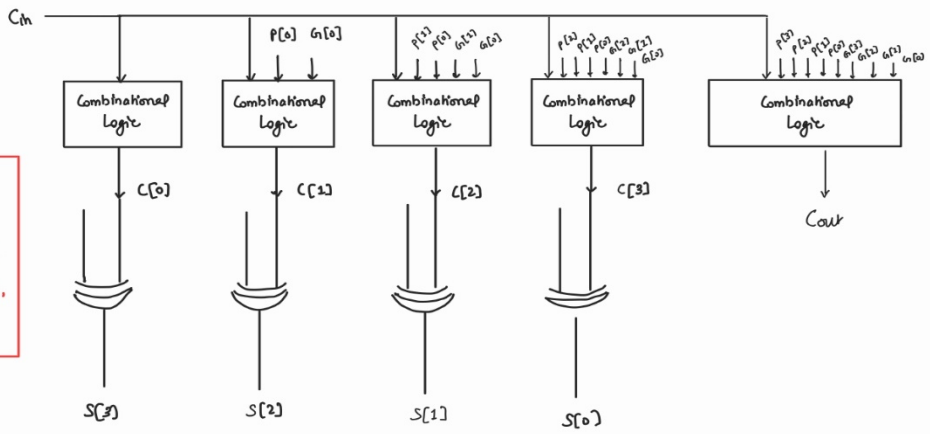
ALU module:

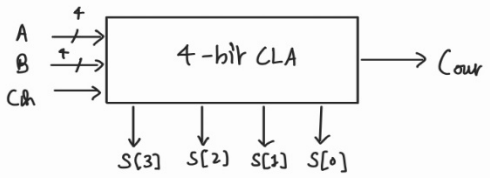
- In the top-level module, **ALU**, we instantiate all the operator modules and assign a parameter to each of the operation after which we use case switch to implement the corresponding module depending upon the input “func” and get the according result.
- The detailed schematic diagrams of each module along with the overall ALU is attached below (in the next pages)

Adder Module :

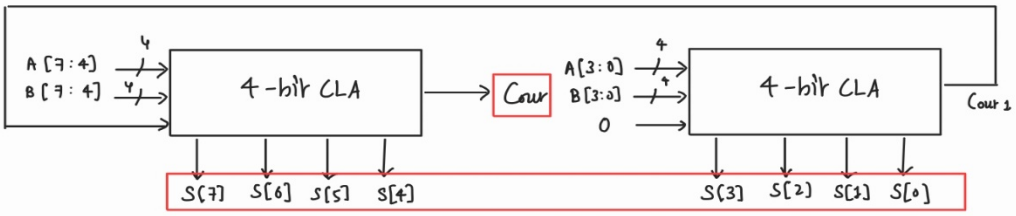
4-bit CLA :

P and G are obtained by bitwise XOR and AND of A and B, respectively

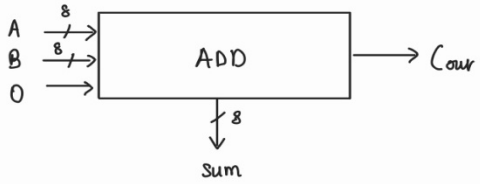




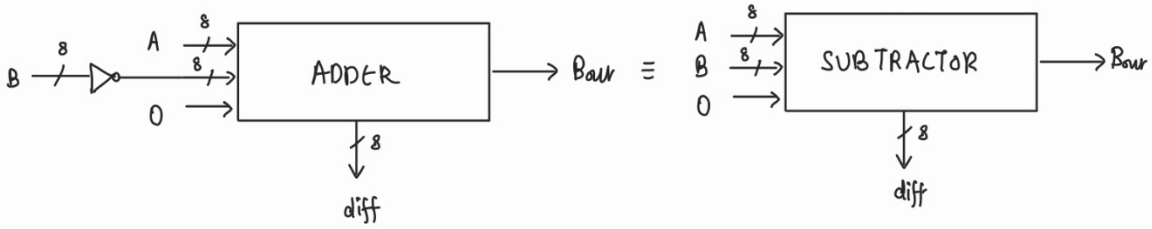
8-bit adder :



Taking Cin = 0 by default



Subtractor Module :

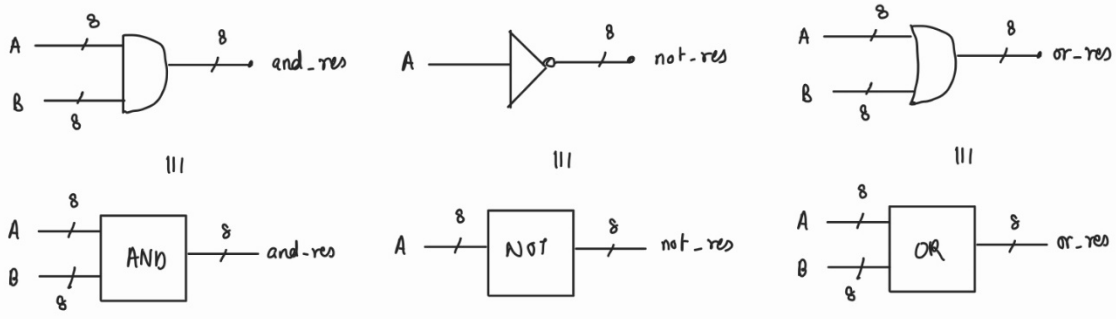


Move Module :

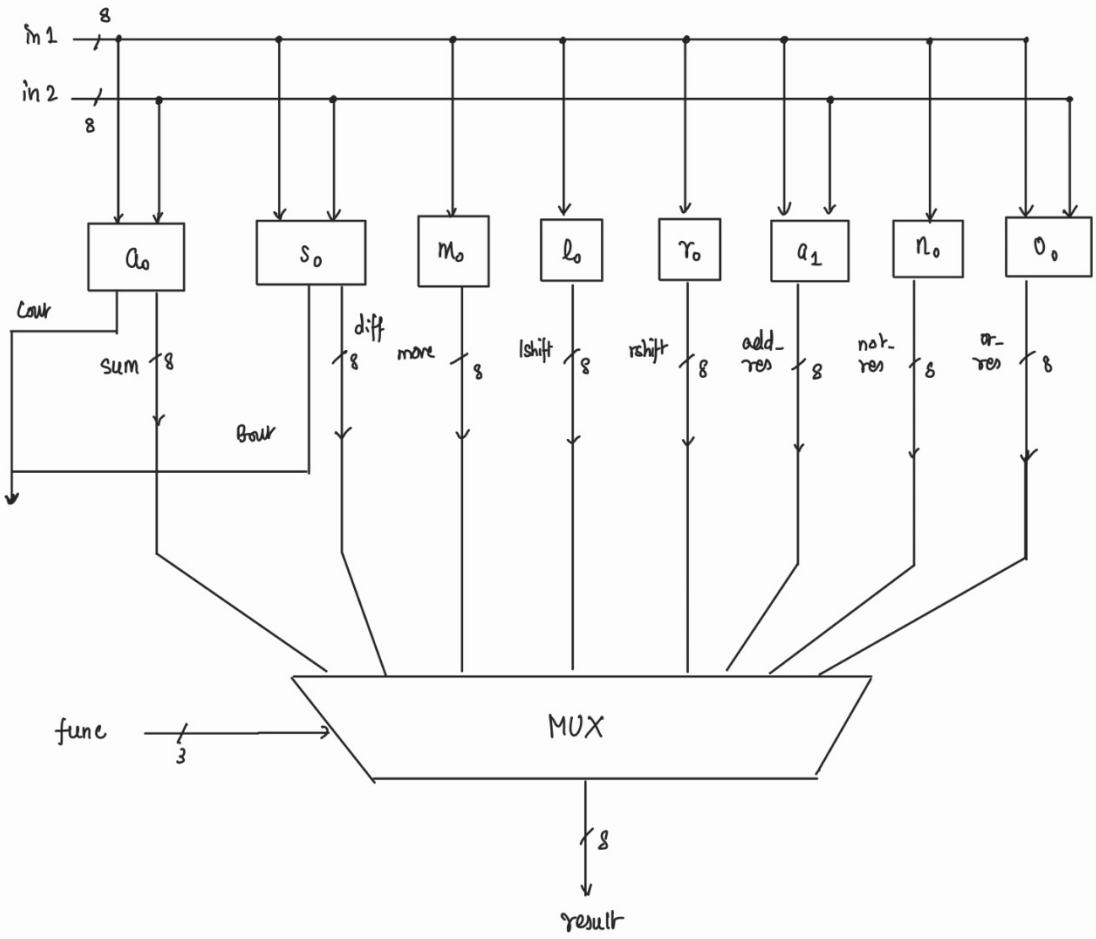
Leftshift & Rightshift Modules :

And, Not & Or Modules :

→ AND, NOT & OR are done for each bit, but for simplicity, is represented as a whole



ALU Module :


in 1
8
in 2
8
a_0
s_0
m_0
l_0
r_0
a_1
n_0
o_0
cout
sum
8
diff
8
more
8
lshift
8
rshift
8
add_res
8
nor_res
8
or_res
8
bout
func
3
MUX
8
result