# CSU22022 Computer Architecture I

## Thirteenth Lecture - Micro-operations

Michael Manzke

2023-2024

Trinity College Dublin

# Logic $\mu$ops

| Micro-ops | Description |
|---|---|
| R0$\leftarrow\overline{R1}$ | Logical bitwise NOT (1's compliment) |
| R0$\leftarrow$R1$\wedge$R2 | Logical bitwise AND (clears bits) |
| R0$\leftarrow$R1$\vee$R2 | Logical bitwise OR (sets bits) |
| R0$\leftarrow$R1$\oplus$R2 | Logical bitwise XOR (complements bits) |

**Table 1:** The aim here is to provide an effective set of bitwise functions. This is a typical basic set.

# $\wedge$ and $\vee$

George Bode Prof. of mathematics in UCC introduced the notation '$\wedge$' and '$\vee$' in 1854, they are used in Register Transfer (RT) notation if it is necessary to distinguish addition from logical OR. For example:

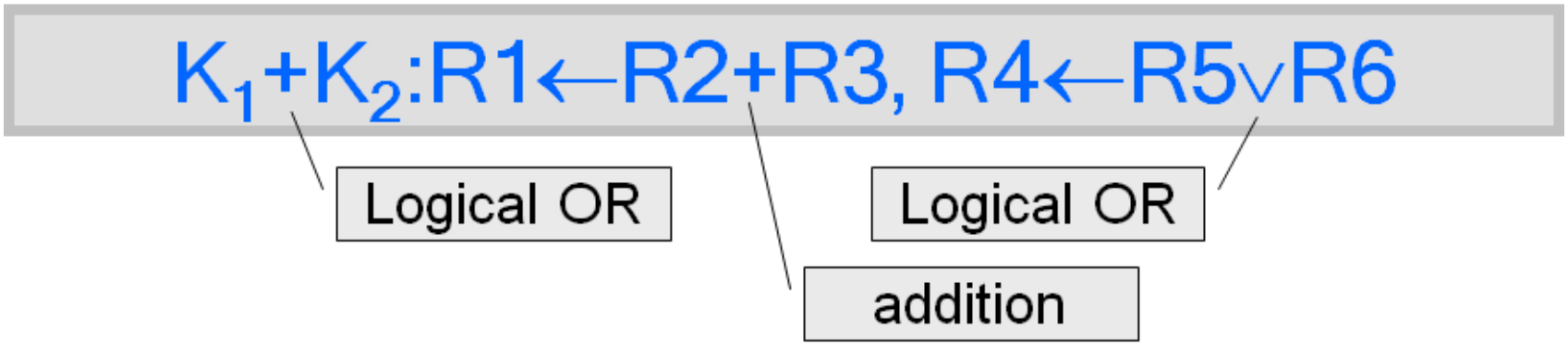



**Figure 1:** The OR micro-ops will always use $\vee$.

# Shift Micro-operations

These provide lateral bitwise shift which are essential for many basic arithmetic algorithms e.g., multiplication division , square root. . .

The minimal set is:

$$R \leftarrow srR \equiv R_i \leftarrow R_{i+1}, i = 0, n-2, R_{n-1} \leftarrow 0 \tag{1}$$

$$R \leftarrow slR \equiv R_i \leftarrow R_{i-1}, i = 1, n-1, R_0 \leftarrow 0 \tag{2}$$

# Logical Shift

These are logical shifts and from them you can develop variants which handle the end bits differently, e.g., arithmetic shift, rotates. . .

# Transfer Micro-operations

- Provide choice of path
- Two approaches are used:
  - Multiplexer-based transfer for speed
  - Bus-based transfers for flexibility and economy
    - We do not use Bus-based transfers in our processor design

# Transfer with Multiplexer

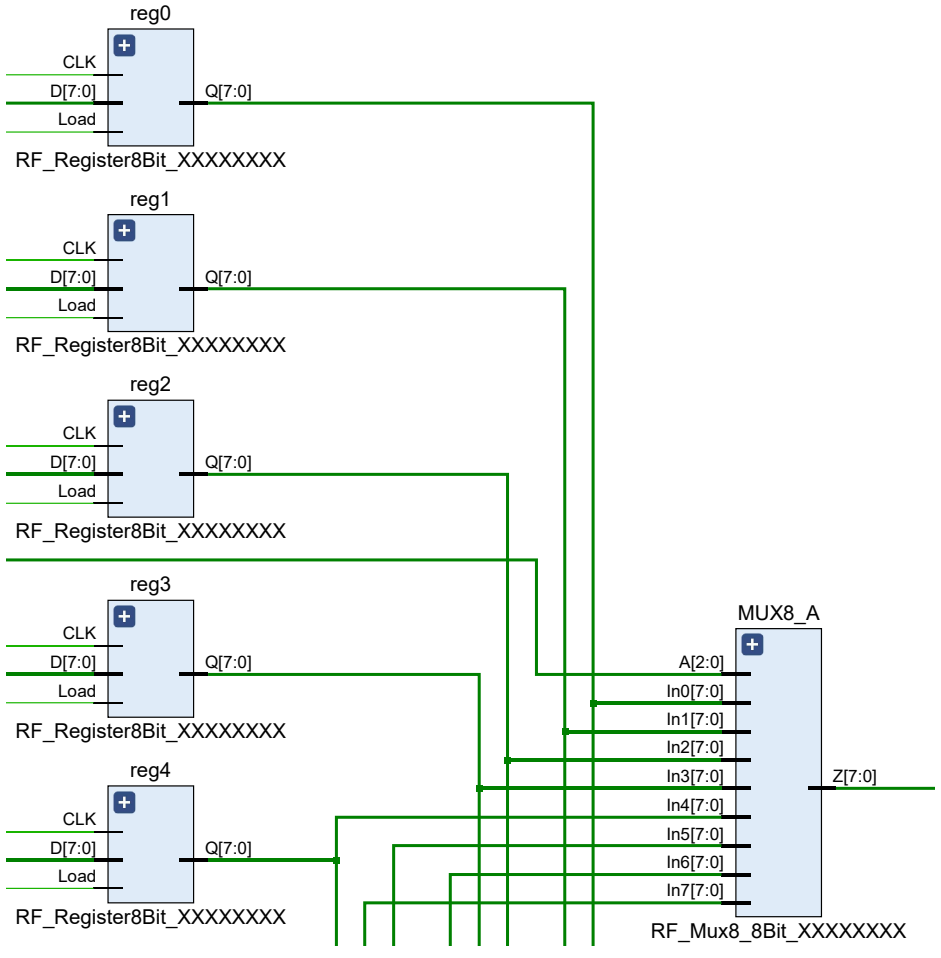
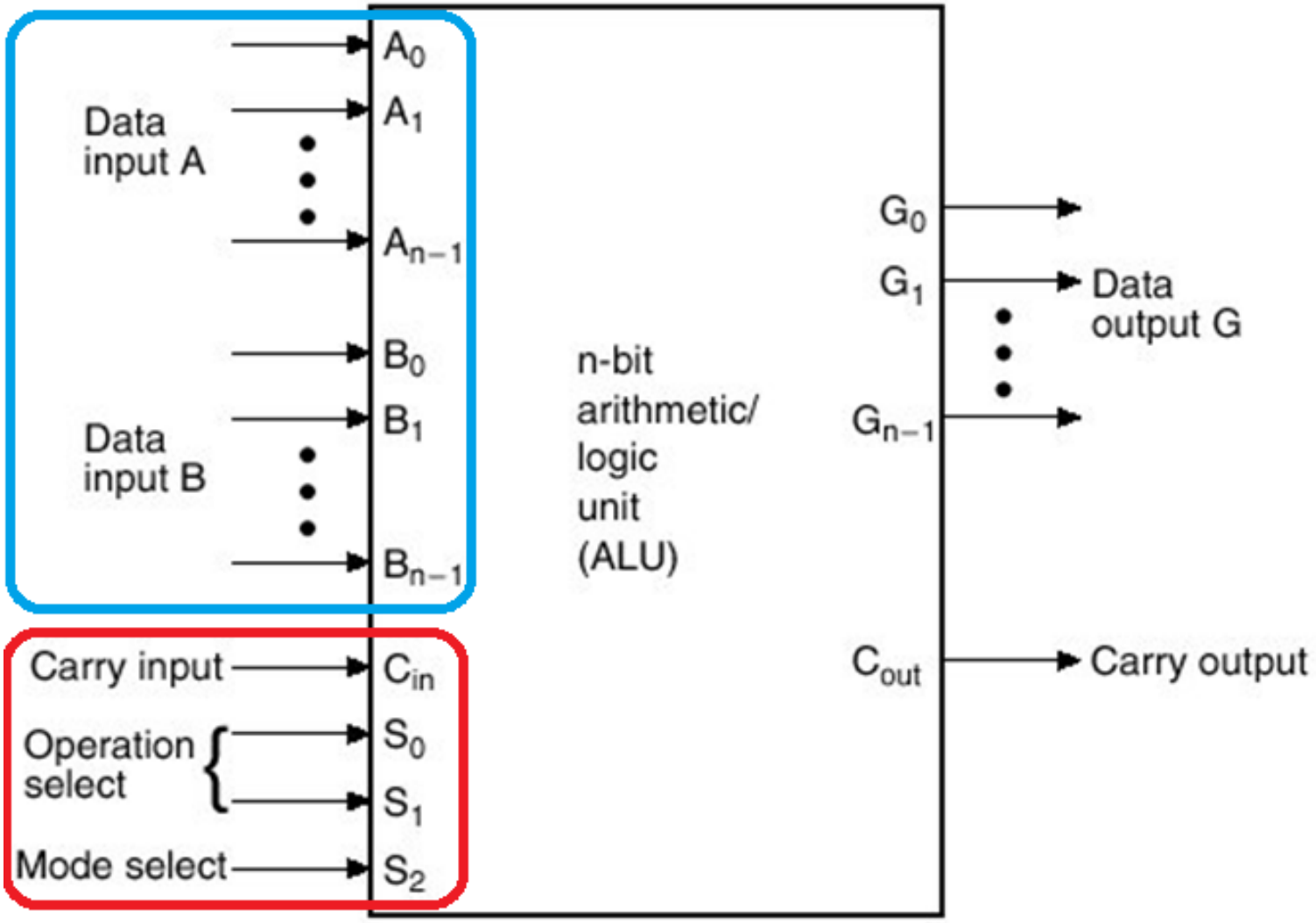



**Figure 2:** The address A[2:0] allows us to select the source register for a transfer to the destination register.

# Arithmetic Circuit

- The arithmetic circuit may be implemented with the following components:
  - Ripple Carry Adder
    from a cascade of full-adder circuits
  - The data input to the parallel adder is manipulated in order to achieve several arithmetic operations

# n-bit Arithmetic Logic Unit(ALU)



# 32-bit Arithmetic Logic Unit(ALU)

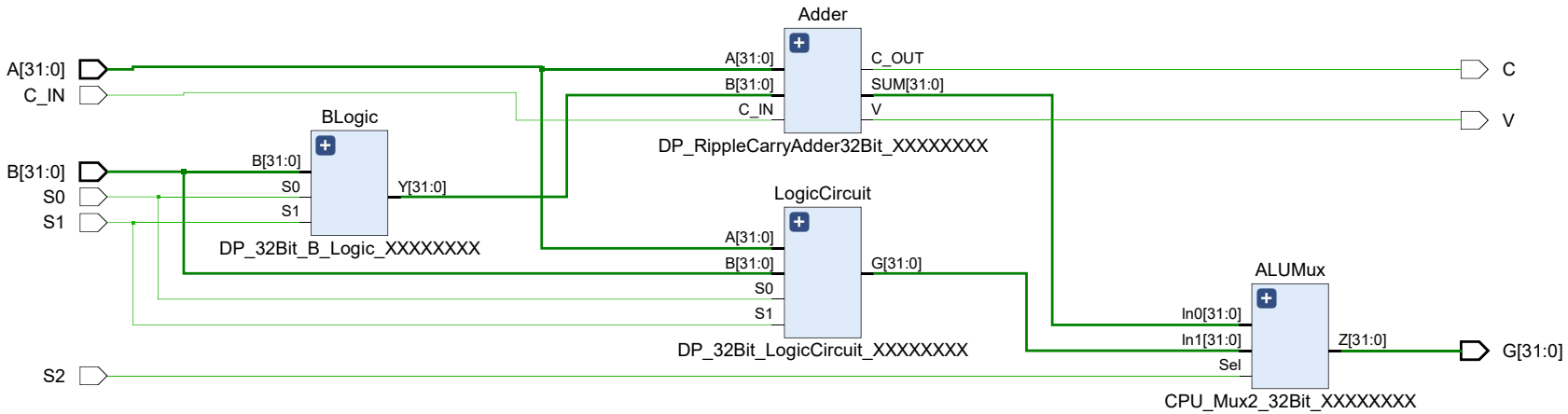



**Figure 3:** The ALU has four input signals $C_{in}, S_0, S_1$ and $S_2$ that determine the operation of the unit. $C_{in}, S_0, S_1$ is used for the BLogic and the Adder. Also, $S_0, S_1$ determines the operation of the LogicCircuit. Both the adder and the LogicCircuit compute an output, but $S_2$ selects one output.

# Signals $S_1, S_0$ select the Blogic Operation

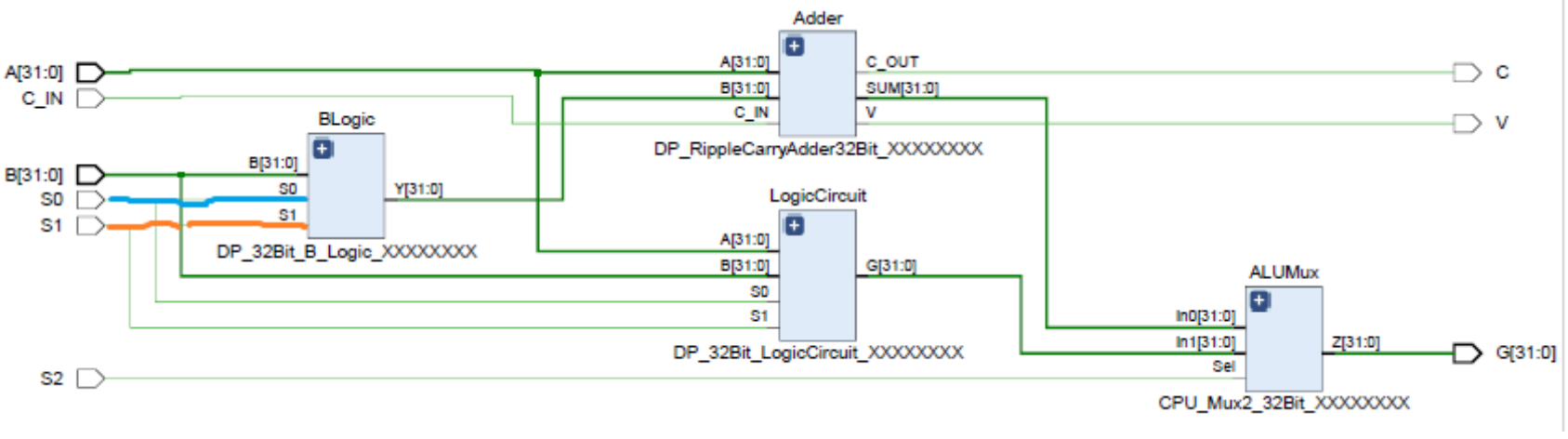



**Figure 4:** Table?? depicts the Blogic Operation

# Signals $S_1, S_0$ select the Logic Circuit Operation

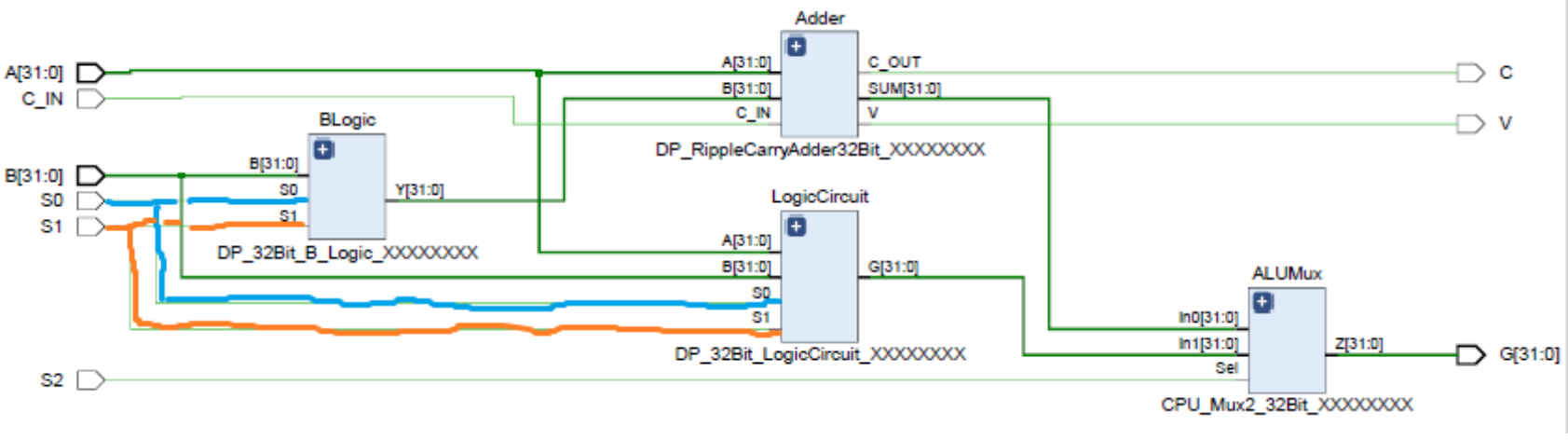



**Figure 5:** The signals $S_1, S_0$ at the same time select the Logic Circuit Operation

# The $S_2$ selects Adder or Logic Circuit

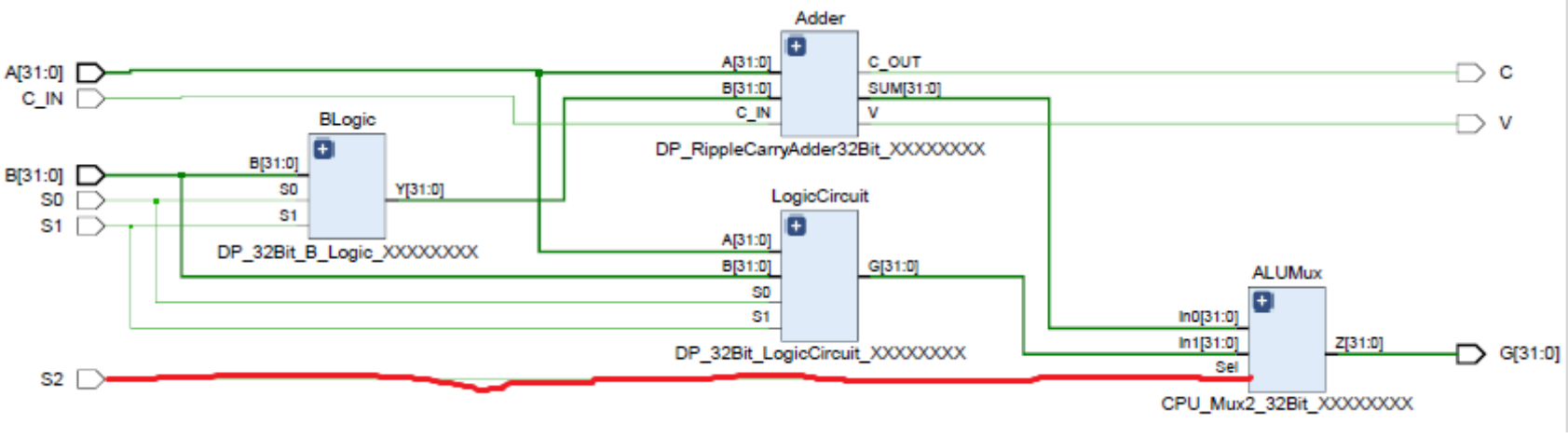



**Figure 6:** The signals $S_2$ selects the desired output, arithmetic or logic.