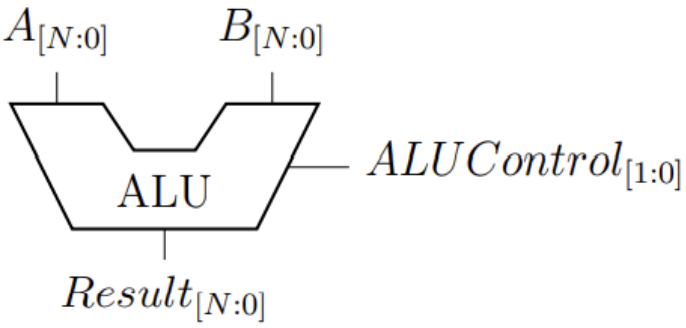
# ALU : Arithmetic logic unit

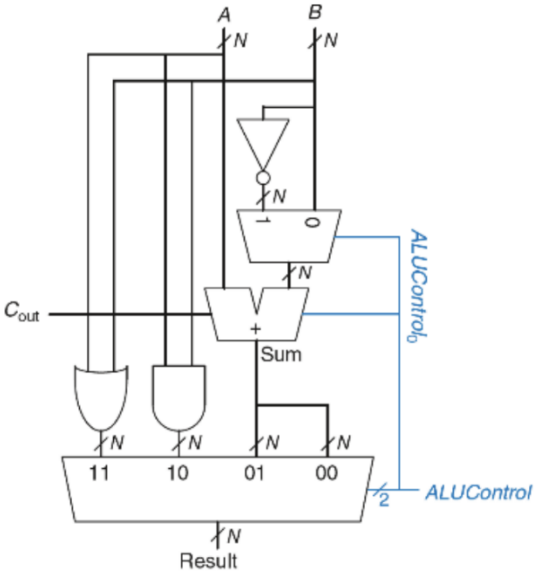
L'**ALU** dovrebbe eseguire 4 operazioni base :

Addizione, Sottrazione, AND, OR.




L'ALU ha dei segnali di controllo, dato che bisogna controllare quale operazione va eseguita, ha poi segnali di ingresso ed uscita. È necessario un circuito combinatorio che realizzi le 4 operazioni sopracitate. Il segnale di controllo, chiamato **ALUControl** seleziona l'operazione. L'ALU inoltre utilizza 4 output aggiuntivi denominati **Flag**, e dato che possono causarsi **Overflow**, servono 2 bit aggiuntivi per gestire tale problema, con o senza segno, servono quindi 2 Flag :

- Overflow (senza segno)
- Carry (con segno)

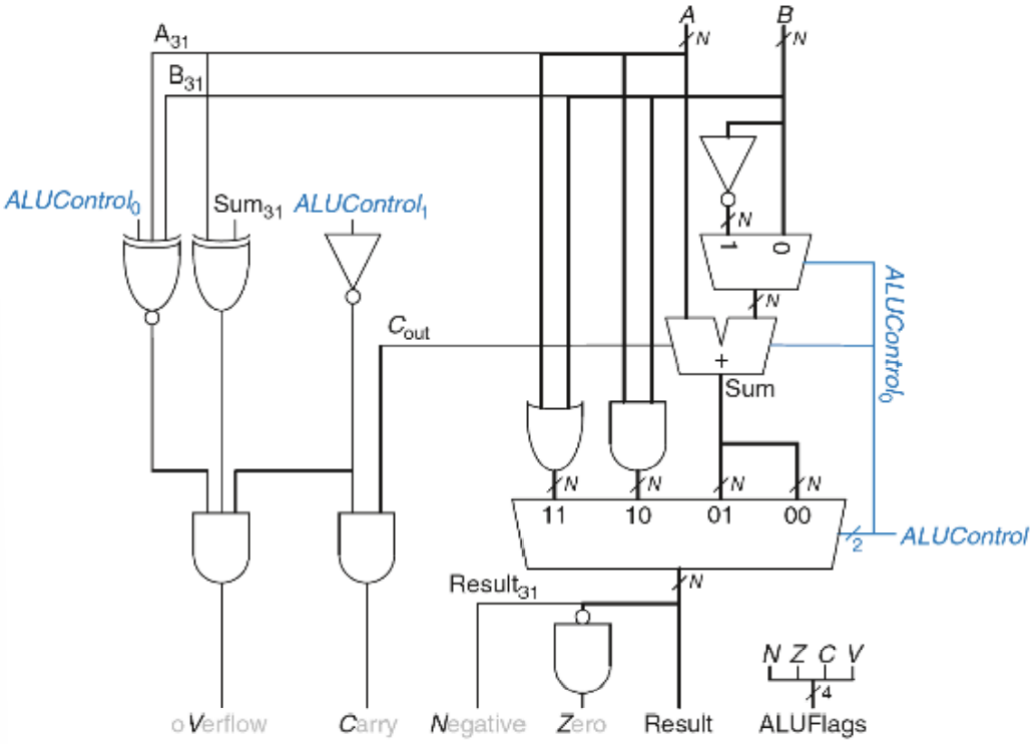



Qui sulla sinistra è mostrato il circuito interno di un ALU. Per le operazioni di AND ed OR abbiamo i gate logici necessari, per l'addizione utilizziamo un **sommatore**, che funzionerà anche per la sottrazione, però avendo la B negata, tale selezione avviene grazie ad un **MUX 2:1**, Controllato dal primo bit del segnale ALUControl. Vi è poi un **MUX 4:1** che ha come scopo quello di selezionare l'operazione che si intende eseguire.

# I Flag di stato

Si è prima parlato dei 4 output aggiuntivi dell'ALU, essi sono chiamati Flag di stato e forniscono informazioni aggiuntive riguardo il risultato dell'operazione :

- N – Il risultato è negativo, quindi il bit più significativo vale 1 (si ricordi il complemento a 2).
- Z – Il risultato è uno zero, quindi tutti i bit valgono zero, si fanno passare attraverso un AND negato, facendo si che il risultato sia 1 se e solo se tutti i bit entrati siano uguali a 0.
- C – Tale Flag controlla se l'operazione ha generato un riporto (carry), ed è controllata dal $C_{out}$ del sommatore messo in AND con il secondo bit dell'ALUControl negato.
- V – Controlla se il risultato ha generato un overflow, è stata quindi svolta una somma tra due numeri di segno uguale (o una sottrazione tra due numeri di segno opposto) in cui il risultato generato è di segno opposto. Viene quindi controllato facendo un AND tra : il secondo bit dell'ALUControl negato, lo XOR tra l'ultimo bit significativo di A e l'ultimo bit significativo del risultato, lo XOR negato tra l'ultimo bit significativo di A, tra l'ultimo bit significativo di B e il primo bit dell'ALUControl.



Esercizio di esempio PLA

Abbiamo 4 variabili $(x_4, x_3, x_2, x_1)$, la funzione $f$ vale 1 quando $x_4 + x_2 x_1 = 0$ e vale $dont\ care$ quando $x_4 x_1 = 1$.

Esiste una seconda funzione $g$ che vale 1 se $x_4 = x_2$.

Rappresentare le due funzioni tramite un circuito PLA, realizzando tabella della verità de mappa di Karnaugh.

| $x_4$ | $x_3$ | $x_2$ | $x_1$ | $f$ | $g$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | x | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | x | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | x | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | x | 1 |