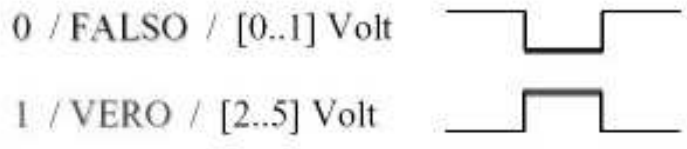
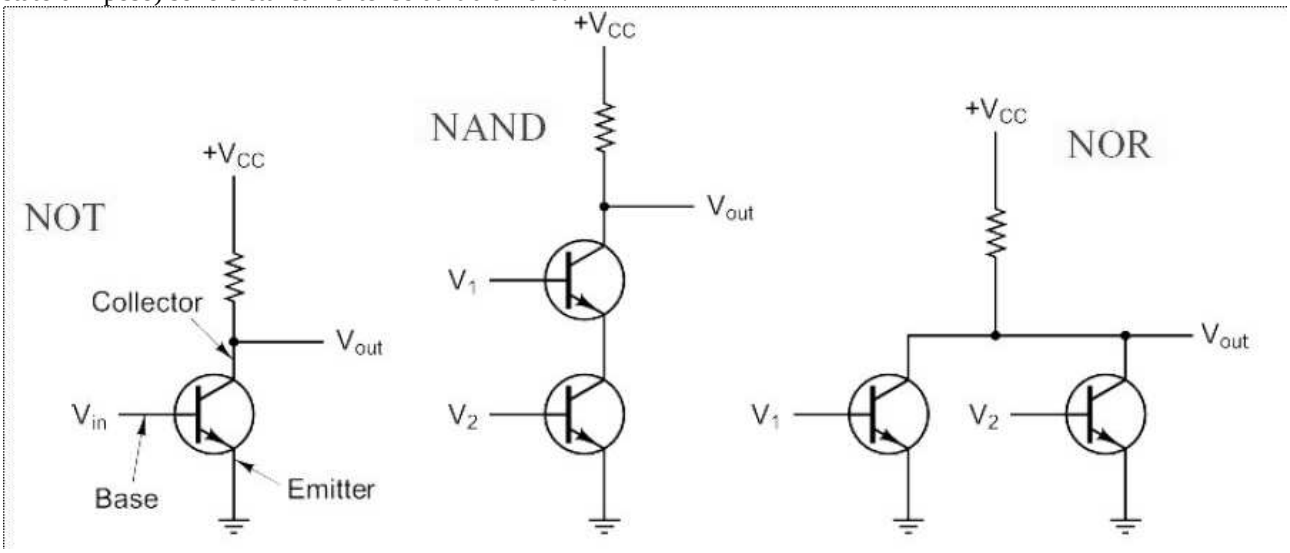
Reti Combinatorie

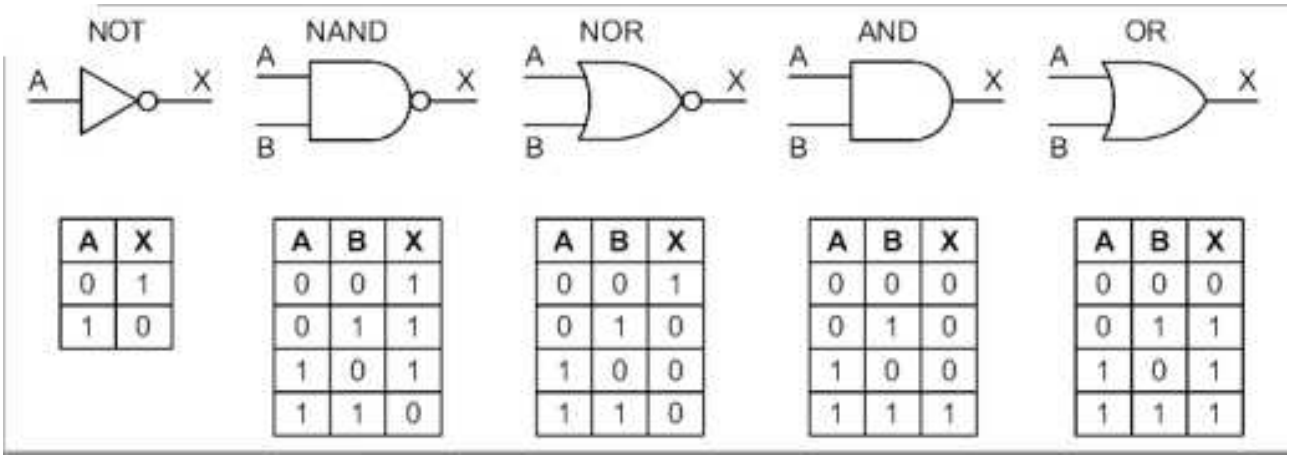
Un calcolatore è costituito da circuiti digitali (hardware) che provvedono a realizzare fisicamente il calcolo. Tali circuiti digitali possono essere classificati in due classi dette reti combinatorie e reti sequenziali.
Le reti combinatorie rappresentano l'implementazione di funzioni 'pure' (ossia **senza stato**) a livello hardware. In particolare, una rete combinatoria implementa una rete logica con **n** ingressi binari ed **m** uscite binarie. Poiché una rete combinatoria implementa una funzione, ad ogni combinazione di valori di ingresso corrisponde **una ed una sola** combinazione di valori di uscita.
In un circuito digitale, i valori binari (vero e falso) sono ottenuti tramite discretizzazione dei segnali. In particolare, nella maggior parte dei calcolatori il valore falso (0) è rappresentato dai segnali con voltaggio nell'intervallo [0,1], mentre i segnali con voltaggio in [2,5] rappresentano il valore di verità vero (1).



Una rete combinatoria è costituita da elementi base, dette porte logiche, che realizzano le operazioni principali dell'algebra Booleana. A loro volta le porte logiche sono implementate tramite elementi detti transistor, che possono essere compresi come interruttori miniaturizzati che si possono controllare elettronicamente. Ciò avviene tramite il segnale Base, che se presente ha l'effetto di porre in comunicazione diretta l'emettitore (emitter) con il collettore (collector) che altrimenti (in stato di riposo) sono elettricamente isolati tra di loro.



Una funzione logica può essere realizzata tramite composizione di porte logiche opportune, che implementano funzioni logiche elementari, come il NOT, il NAND, il NOR, l'AND, e l'OR.



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

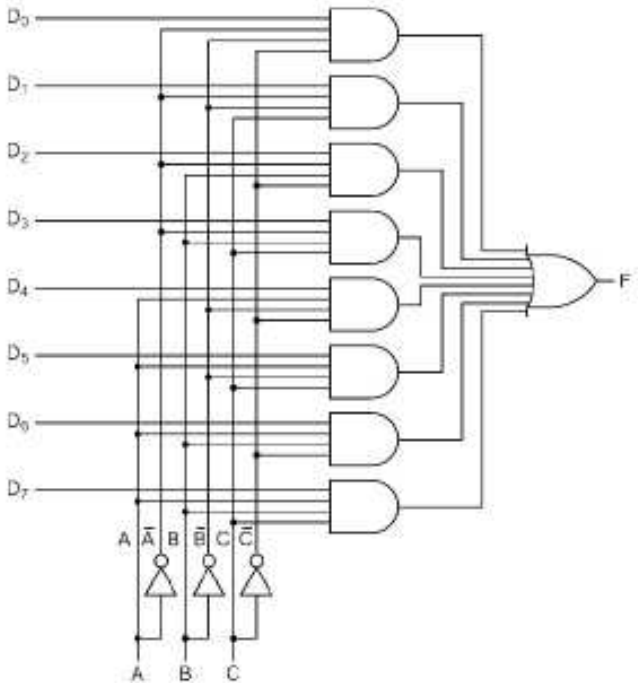
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

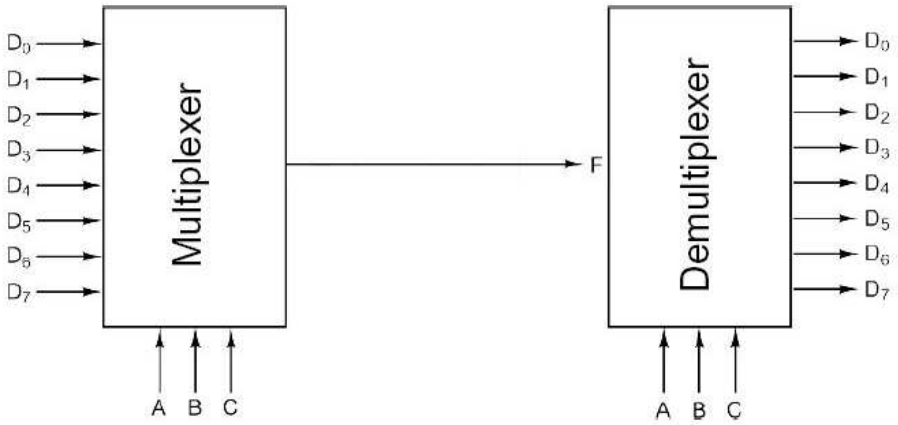
Per facilitare la costruzione di reti combinatorie complesse, tuttavia, è preferibile comporre reti combinatorie predefinite piuttosto che partire dalle porte logiche.
Alcuni esempi tipici di reti combinatorie predefinite sono ad esempio il confrontatore, il commutatore, il selezionatore.
Un confrontatore, a due ingressi (x,y) ed una uscita (z), è definito come z := **not** (x = y);
Un commutatore, a due ingressi primari (x,y), un ingresso di controllo (α) ed una uscita (z), è definito come z := **if not** α **then** x **else** y; Un selettore, ad un ingresso primario (x), un ingresso di controllo (α) e due uscite (z_1,z_2) è definito come **if not** α **then (**z_1 := x ; z_2 := 0) **else** (z_1 := 0 ; z_2 := x).
Dalle specifiche di sopra, si può evincere che, in generale, una rete combinatoria è costituita da m ingressi primari (x_1,…, x_m), n ingressi di controllo (α_1,…, α_n), una uscita (z):

z := **case** α_1 α_2 … α_n
when 00…0 => x_1
when 00…1 => x_2
…
when 11…1 => x_m
end case

Vediamo alcune possibili implementazioni di reti combinatorie di interesse. Di seguito è riportato un circuito che

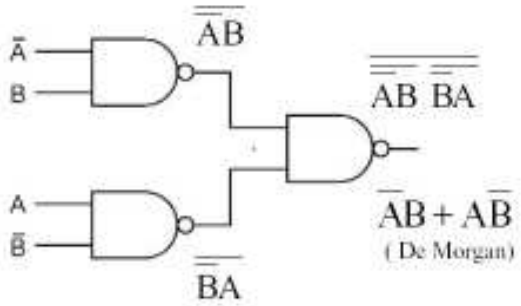


implementa un Multiplexer, che risolve il problema di portare verso l'esterno i segnali di un circuito usando solo poche "vie di uscita" (piedini di un chip). In generale, un Multiplexer ha 2^n linee di ingresso (nella figura si hanno $2^3 = 8$ linee di ingresso denominate D_0 - D_7); n linee di controllo (nella figura, 3 linee di controllo denominate A,B,C), ed 1 linea di output (nella figura, denominata F). Ogni possibile configurazione dei bit di controllo seleziona una delle linee di ingresso e quindi permette al segnale che percorre tale linea di essere trasferito alla linea di uscita F. Ad esempio, se i segnali di controllo sono A=1, B=0, e C=1, viene selezionata la linea di ingresso D_5, infatti tale configurazione dei bit di controllo permette solo alla porta AND in cui la linea D_5 e' collegata di "trasferire l'informazione all'uscita F. Questo perché quando sulla linea si presenta il segnale 0, anche l'uscita dell'AND sarà 0, e quando si presenta il segnale 1, essendo tutti gli ingressi della porta AND tutti a 1, anche l'uscita dell'AND sarà 1. Al contrario, tutte le altre porte AND restituiranno sempre valore 0 in quanto almeno uno degli ingressi (o meglio dei bit di controllo, eventualmente negati) sarà sicuramente a 0.
Un altro esempio di rete combinatoria utile è il circuito inverso del Multiplexer, detto Demultiplexer, che è spesso usato in combinazione con quest'ultimo in quanto permette di direzionare un segnale in ingresso verso una delle linee di uscita.
Tale circuito è ad esempio utilizzato per far "entrare" segnali con destinazioni diverse all'interno di un circuito integrato attraverso un singolo piedino.

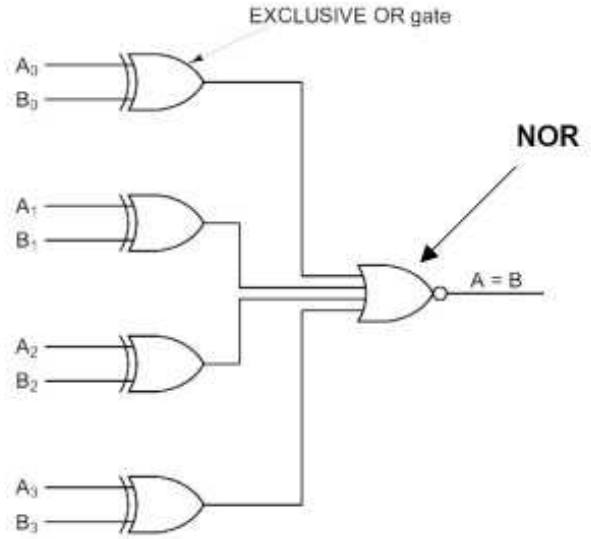


Un comparatore molto semplice per un singolo bit puó essere realizzato tramite una porta logica XOR.

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

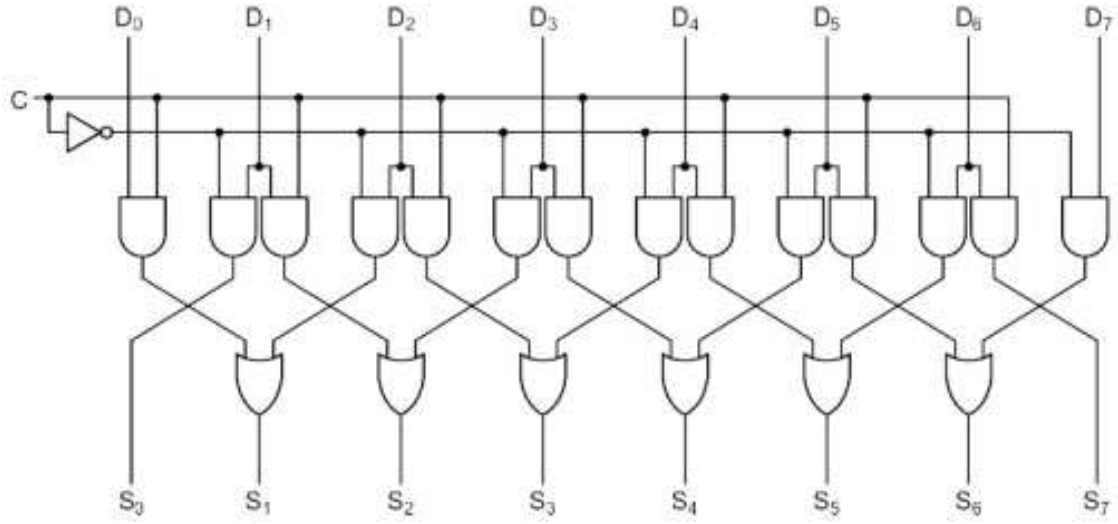


Un circuito che implementa un comparatore per piú bit è realizzabile connettendo piú porte XOR tramite una porta NOR:



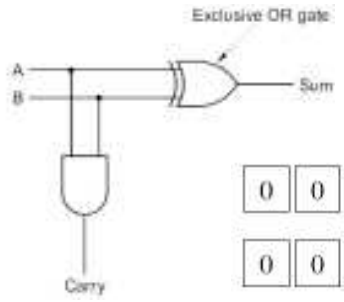
Infatti, il NOR restituisce valore di veritá vero (cioè 1) quando tutti i suoi ingressi sono falsi (cioè 0), quindi quando tutti i bit A_i sono identici ai bit B_i.

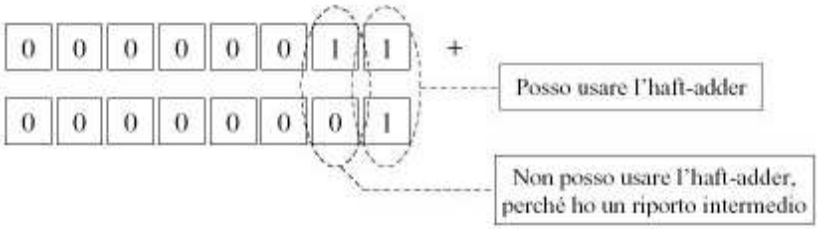
Altre reti combinatorie predefinite molto utili sono quelle che realizzano operatori aritmetico logici a specifica **diretta**, come la addizione, la sottrazione, la traslazione, la rotazione, l'incremento, il decremento, etc.
Alcuni di tali reti combinatorie, tuttavia, per motivi di economia nell'uso di porte logiche, sono tipicamente implementate congiuntamente in una unica rete aritmetico logica **multi-funzione** che eseguono **una** delle operazioni suddette a seconda del valore assunto da un certo numero di variabili di controllo. Le reti multi-funzione si usano per implementare le **ALU** (*arithmetic logic unit*), che costituiscono i circuiti all'interno della unitá di calcolo centrale predisposti alla esecuzione delle operazioni aritmetico logiche.
Di seguito si mostra un esempio di circuito per la realizzazione di un traslatore (shifter). Come si vede dalla figura, questo è



realizzato tramite piú porte logiche AND che permettono, in base al valore del bit di controllo C, la traslazione della configurazione di bit in ingresso D a destra (C=1) o a sinistra (C=0). Si noti come il nuovo bit inserito a causa della traslazione (a destra o a sinistra) abbia sempre valore 0.
La addizione, invece, viene realizzata attraverso la composizione di circuiti che operano sui singoli bit della rappresentazione binaria dei numeri da sommare. Un primo esempio di circuito sommatore per un bit è dato dall'Half-Adder mostrato di seguito:

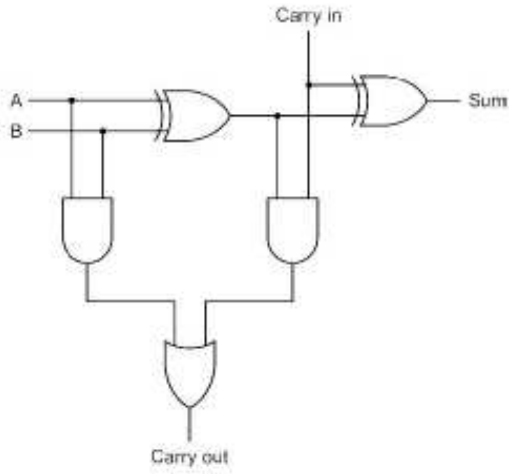
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



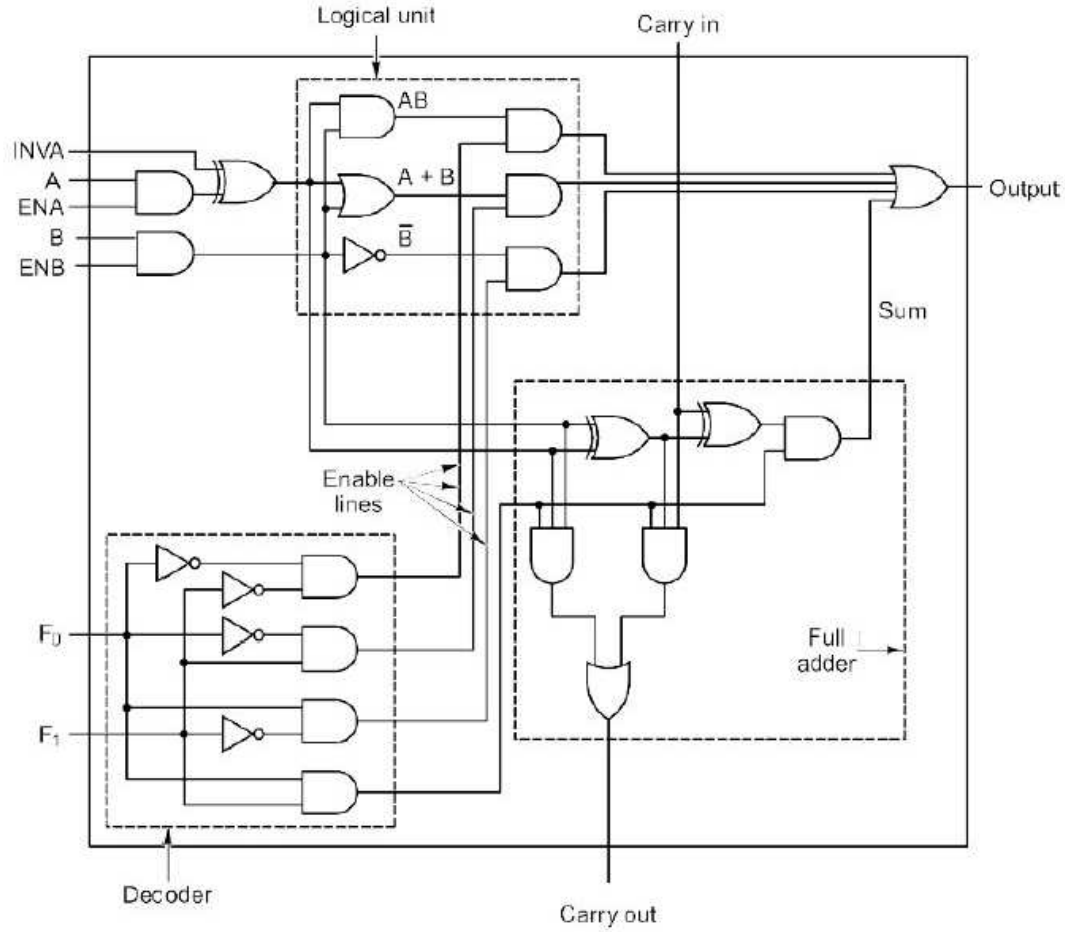


Un problema dell'Half-Adder è che questo non puó essere usato per effettuare la somma in parallelo di piú bit che rappresentano i due numeri da addizionare. Infatti, l'Half-Adder non permette di considerare l'eventuale bit di riporto che si genera nella addizione di due bit. Questo problema è risolto dal Full-Adder che prevede un input aggiuntivo per "ospitare" l'eventuale riporto della addizione sul bit precedente:

A	B	Carry in	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Utilizzando n Full-Adder è possibile realizzare la somma di numeri rappresentati con n bit. Ovviamente la linea di riporto di ogni Full-Adder (Carry Out) deve essere collegato alla linea di riporto in ingresso (Carry In) al successivo Full-Adder. Cioè il Carry Out del bit i-esimo deve essere collegato al Carry In del bit i+1-esimo (i=0,…,n-2).
Di seguito è mostrata una ALU ad 1 bit che realizza 4 operazioni, selezionate dai bit di controllo F_0 e F_1.

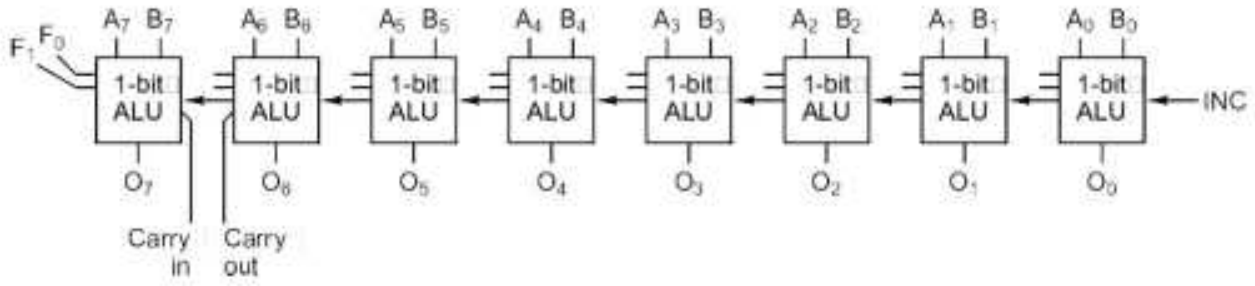


Le operazioni realizzate dalla ALU mostrata in figura sono le 4 operazioni base che tipicamente ogni ALU implementa:

- **AND** logico bit a bit (A **AND** B);
- **OR** logico bit a bit (A **OR** B);
- Complemento o negazione;
- Somma aritmetica (A+B).

Le operazioni logiche AND, OR e negazione sono implementate in alto, nella parte denominata unitá logica. La somma è invece realizzata in basso a destra tramite un full-adder a 1 bit, diversa da quella vista in precedenza in quanto alcune porte logiche AND sono utilizzate per abilitare o meno le uscite nel caso in cui la somma sia selezionata tramite i bit di controllo F_0 e F_1. Tali bit di controllo sono decodificati da un decoder in basso a sinistra. Tale decoder trasforma i due bit di controllo in 4 linee di abilitazione, una per ogni operazione che la ALU puó eseguire. Gli ingressi ENA e ENB possono essere utilizzati per forzare a 0 gli input A e B, mentre impostando a 1 la linea INVA l'input A viene invertito. Infine notare che una sola delle 4 funzioni che la ALU puó calcolare viene portata in output tramite la porta OR.

Componendo opportunamente piú ALU ad un bit si puó ottenere una ALU a n bit. Di seguito è mostrata ad esempio una ALU ottenuta in questo modo:



I bit di controllo F_0 e F_1 sono collegati a tutte le ALU a 1 bit, mentre il bit di riporto (Carry out) di una ALU è collegato all'ingresso di riporto (Carry in) della successiva. Notare che il riporto non si propaga istantaneamente ma subisce un ritardo. Quindi puó accadere che un riporto si possa propagare dalla ALU associata al bit meno significativo fino alla ALU associata al bit piú significativo. In questo caso il ritardo è di n=8 unitá di tempo. Infine si noti l'ingresso aggiuntivo INC che permette di sommare 1 al risultato. Ció si ottiene molto semplicemente collegando la linea INC all'ingresso di riporto (Carry in) della ALU associata al bit meno significativo.