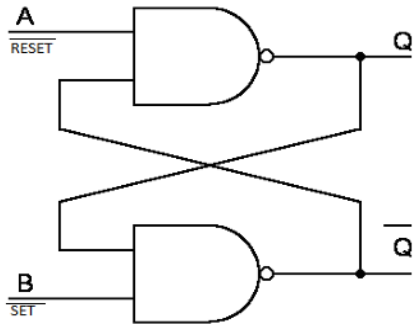
I flip flop sono circuiti digitali sequenziali che hanno il compito di memorizzare un bit. Un circuito digitale si dice sequenziale se l'uscita dipende dagli ingressi applicati e dallo stato precedente della stessa uscita. Un circuito sequenziale , pertanto, deve ricordare il suo stato precedente e quindi deve possedere uno o più elementi di memoria .



A B	Qn+1
1 1	Qn
1 0	0
0 1	1
0 0	?

I flip-flop sono noti, anche, come multivibratori bistabili perché ciascuno degli stati logici 0 e 1 può essere stabile nel tempo

Flip-flop SR (Set-Reset)

Il più semplice dispositivo di memoria è il flip-flop Set-Reset. Esso possiede due ingressi denominati Set e Reset ed una uscita indicata con Q.

I circuiti digitali che realizzano il flip flop sono dotati, spesso, anche dell'uscita Q negata. Occorre precisare, inoltre, che in un dispositivo di memoria, l'uscita dipende non solo dalla particolare combinazione assunta dalle variabili di ingresso ma anche dallo stato precedente assunto dall'uscita Q. Tale stato precedente verrà indicato con Qo.

Flip-flop a porte NAND

Qui gli ingressi sono indicati con le lettere A e B che si comportano come ingressi di S (Set = porta alto) ed R (Reset = azzera) sono a logica negativa, cioè sono attivi bassi (ciò è indicato dal trattino sopra alle due scritte Set e Reset).

In altre parole : A = S , B = R

Ponendo A B=1 0 si realizza la funzione di reset per cui l'uscita Q si porta a 0.

Ponendo A B=0 1 si realizza la funzione di set per cui l'uscita Q si porta a 1.

Ponendo A B=1 1 si realizza la funzione di memoria per cui l'uscita conserva il precedente valore memorizzato.

A B=0 0 è la combinazione da evitare per incongruenza logica.

Quindi possiamo dire che il flip flop RS è un elemento di memoria che riesce a memorizzare solamente 1 bit ! al suo piedino Q; per memorizzare il bit basta agire sul pin Set o Reset.

Un **multiplexer (MUX)** è un componente elettronico digitale che seleziona uno tra molti segnali di ingresso e lo invia all'uscita in base al valore di uno o più **segnali di selezione**. È fondamentalmente un "commutatore" che permette di scegliere quale ingresso (tra molti) deve essere trasmesso.

Un **MUX** prende più linee di ingresso (tipicamente 2^n ingressi) e ne seleziona una per inviarla all'uscita, usando un numero di **linee di selezione** che determinano quale ingresso attivare.

Esempio: MUX 2:1

Un **MUX 2:1** ha due ingressi (A e B), una linea di selezione (S), e un'uscita (Y):

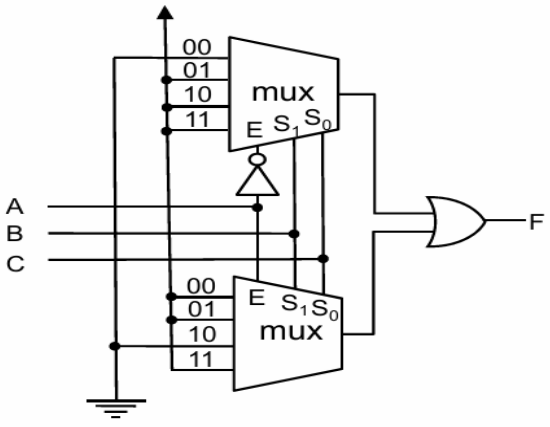
- Se **S = 0**, l'uscita **Y** sarà uguale a **A**.
- Se **S = 1**, l'uscita **Y** sarà uguale a **B**.

Esempio: MUX 4:1

Un **MUX 4:1** ha quattro ingressi (A, B, C, D), due linee di selezione (S1, S0), e una sola uscita:

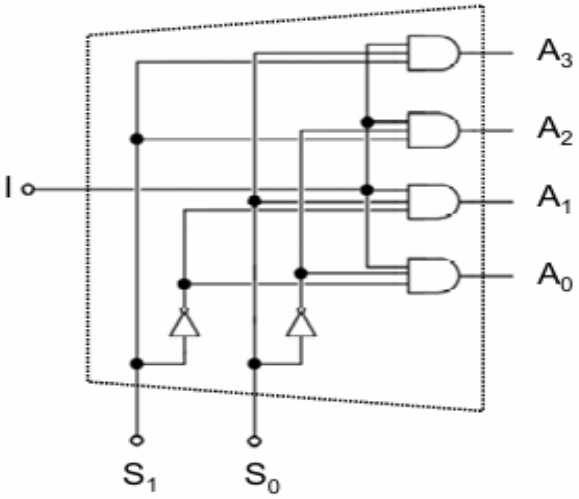
- Se **S1S0 = 00**, l'uscita sarà **A**.
- Se **S1S0 = 01**, l'uscita sarà **B**.
- Se **S1S0 = 10**, l'uscita sarà **C**.
- Se **S1S0 = 11**, l'uscita sarà **D**.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



In elettronica digitale, un **demux** (demultiplexer) è un circuito combinatorio che prende un singolo segnale in ingresso e lo indirizza su una delle molteplici uscite, a seconda di un valore di controllo. In pratica, il demux riceve un flusso di dati e lo distribuisce a più destinazioni, permettendo una gestione efficiente delle risorse nel sistema.

Un **demux** è simile ad un **multiplexer (mux)**, ma con la differenza che quest'ultimo unisce più segnali in un singolo flusso, mentre il demux separa un flusso di dati in più linee. Il demux utilizza un insieme di **linee di selezione** (solitamente in binario) per determinare quale delle uscite deve ricevere il dato.



$$A_3 = I \cdot S_0 \cdot S_1$$

$$A_2 = I \cdot S_0 \cdot \overline{S_1}$$

$$A_1 = I \cdot \overline{S_0} \cdot S_1$$

$$A_0 = I \cdot \overline{S_0} \cdot \overline{S_1}$$

I	S_0	S_1	A_0	A_1	A_2	A_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$$A_3 = I \cdot S_0 \cdot S_1$$

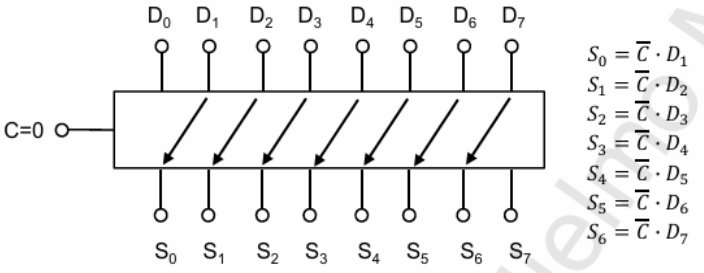
$$A_2 = I \cdot S_0 \cdot \overline{S_1}$$

$$A_1 = I \cdot \overline{S_0} \cdot S_1$$

$$A_0 = I \cdot \overline{S_0} \cdot \overline{S_1}$$

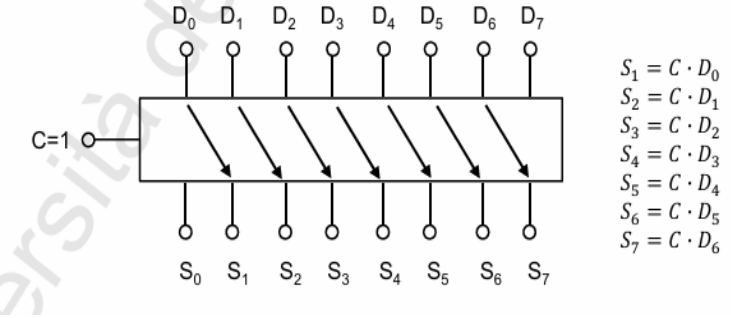
Left Shifter (Spostamento a sinistra):

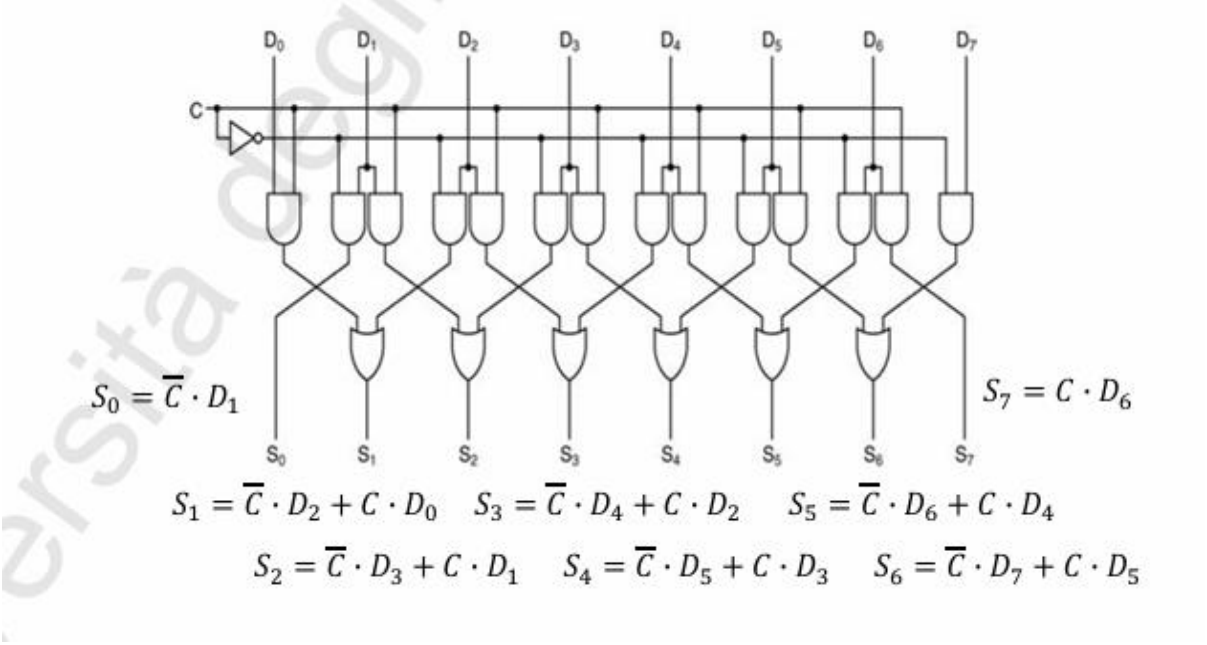
Lo spostamento a sinistra di un segnale su 8 bit significa che ogni bit dell'ingresso viene spostato di una certa quantità di posizioni verso sinistra. Le **nuove posizioni** a destra vengono riempite con **zeri**.



Right Shifter (Spostamento a destra):

Lo spostamento a destra di un segnale su 8 bit comporta che i bit vengano spostati verso destra, con i **bit vuoti** a sinistra che vengono riempiti con **zeri** (nel caso di un right shift logico) o con il **bit di segno** (nel caso di un right shift aritmetico).

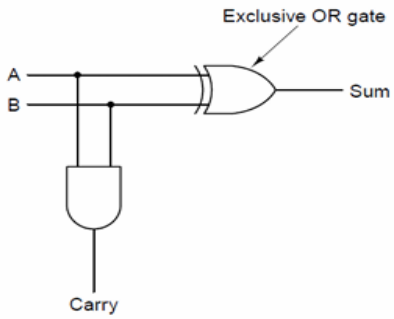




- Un half adder è un circuito in grado di eseguire la somma tra due bit (A e B).
- Può essere impiegato per sommare la cifra meno significativa di due numeri in quanto non considera un eventuale riporto dalle cifre precedenti:

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

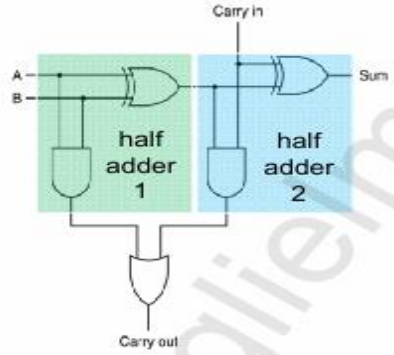
- Il circuito corrispondente è:



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

- Lo schema circuitale è semplice e modulare:

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



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

$$Sum = \overline{A}(\overline{B}C_{in} + B\overline{C_{in}}) + A(\overline{B} \cdot \overline{C_{in}} + BC_{in})$$

$$C_{out} = \overline{A}BC_{in} + A(B+C_{in})$$

↑ Full hadd

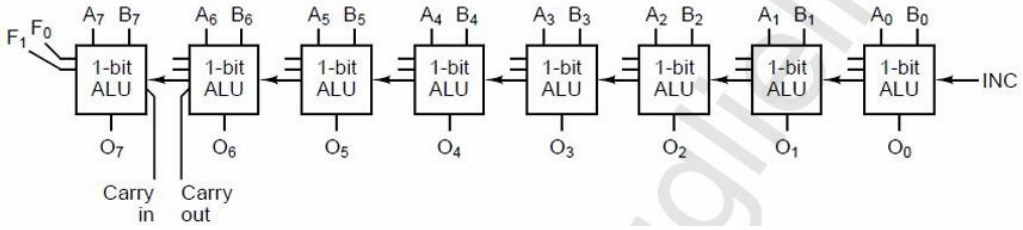
- L'unità ALU contiene 3 differenti circuiti: un decoder, una unità logica e un full adder.
- Il decoder seleziona l'operazione richiesta in base ai segnali F_0 e F_1.

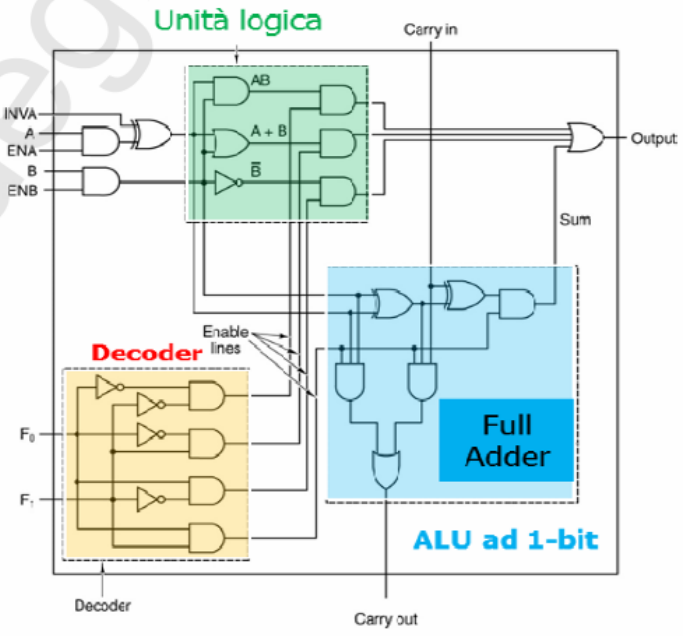
F_0	F_1	Risultato
0	0	A and B
0	1	A or B
1	0	not B
1	1	A + B + Carry in

- Il full adder somma A, B e il riporto in ingresso (Carry in) e calcola il risultato e l'eventuale riporto (Carry out).

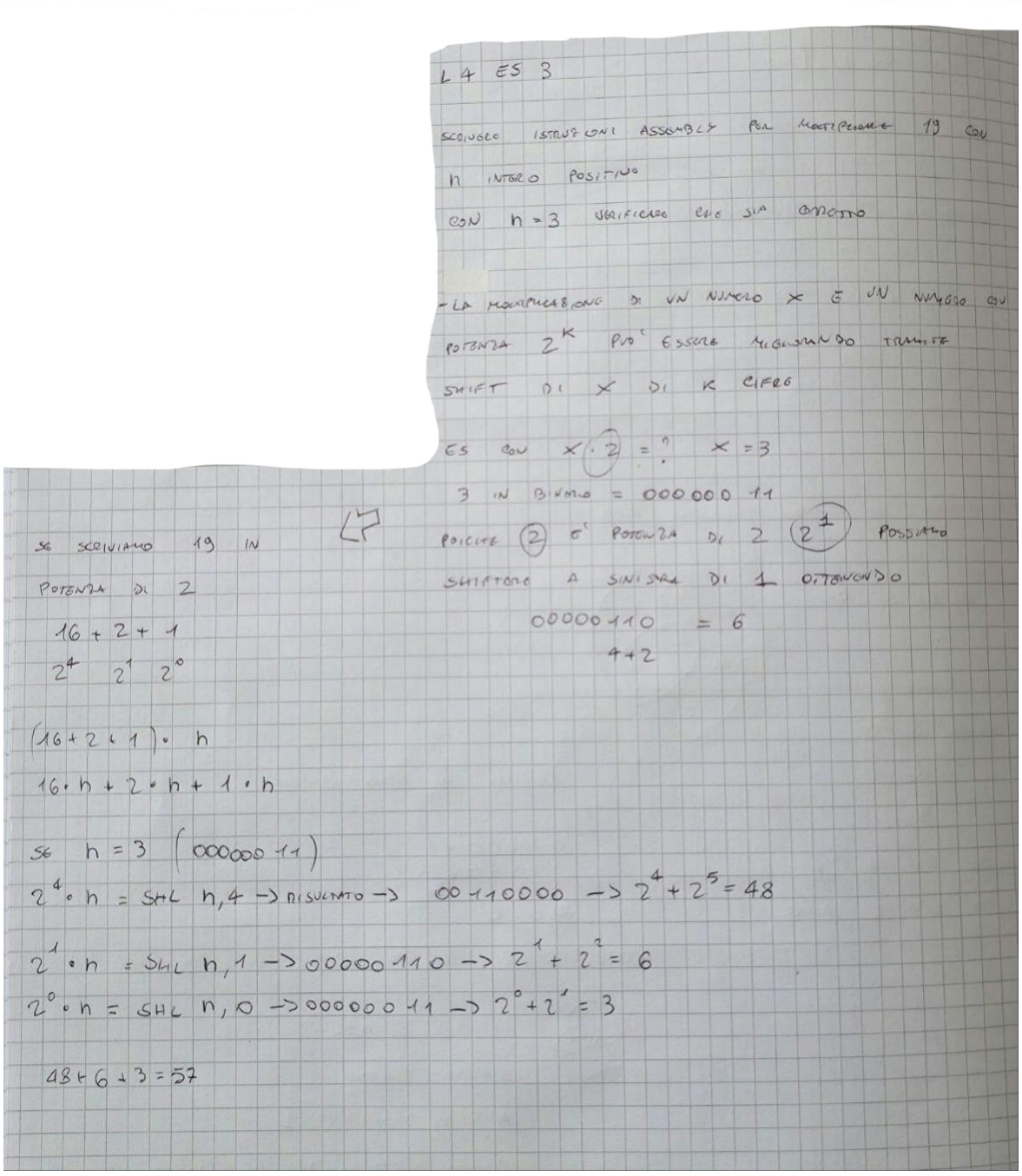
- Le ALU ad 1-bit possono essere assemblate in cascata per costruire ALU di lunghezza variabile.
- Occorre trasferire il riporto dalla cifra meno significativa a quella più significativa e propagare parallelamente i segnali F_0 e F_1.

- Questa tecnica è detta **bit slice** (suddivisione di bit) e può essere applicata a tutti i circuiti digitali che lavorano bit-a-bit.





Moltiplicazioni veloci in assembly con lo shifter



L4 ES 3

SCRIVERE ISTRUZIONI ASSEMBLY PER MOLTIPLICARE 19 CON n INTERO POSITIVO

CON n = 3 VERIFICARE CHE SIA CORRETTO

- LA MOLTIPLICAZIONE DI UN NUMERO x È UN NUMERO CON POTENZA 2^K PUÒ ESSERE MOLTIPLICANDO TRAMITE SHIFT DI x DI K CIFRE

ES CON $x \cdot 2 = ?$ $x = 3$

3 IN BINARIO = 000000 11

POICHÉ (2) È POTENZA DI 2 (2^1) POSSIAMO SHIFTARE A SINISTRA DI 1 OTTENENDO

00000110 = 6

4+2

SE SCRIVIAMO 19 IN POTENZA DI 2

16 + 2 + 1

2^4 2^1 2^0

$(16+2+1) \cdot n$

$16 \cdot n + 2 \cdot n + 1 \cdot n$

SE n = 3 (000000 11)

$2^4 \cdot n$ = SHL n,4 -> RISULTATO -> 00110000 -> $2^4 + 2^5 = 48$

$2^1 \cdot n$ = SHL n,1 -> 00000110 -> $2^1 + 2^2 = 6$

$2^0 \cdot n$ = SHL n,0 -> 00000011 -> $2^0 + 2^1 = 3$

48 + 6 + 3 = 57