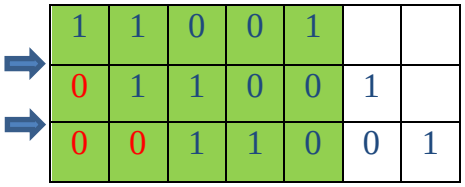
Shifter

Lo **shifter logico** sposta i valori a sinistra o destra, riempendo gli spazi vuoti con lo 0, viene utilizzato per i numeri binari. *Shiftando* a destra un valore, si ottiene il risultato di esso diviso per 2, *shiftandolo* a sinistra, si ottiene il risultato di esso moltiplicato per 2.

Shiftare a destra:

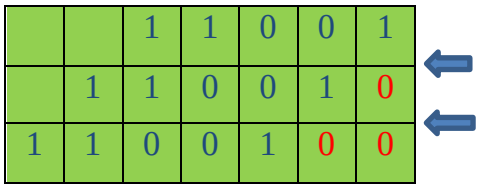
$11001 \gg 2 = 00110$ shiftare 2 volte il valore a destra (dividere il numero per 4)



$11001 = 25 - diviso\ 4 \rightarrow 00110 = 6$ (In questo caso stiamo escludendo la parte decimale)

Shiftare a sinistra:

$11001 \ll 2 = 00110$ shiftare 2 volte il valore a sinistra (moltiplicare il numero per 4)

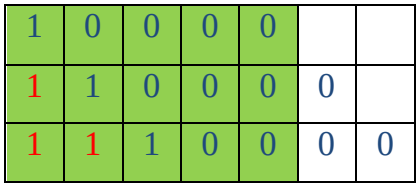


$11001 = 25 - per\ 4 \rightarrow 1100100 = 100$

Lo **shifter aritmetico** si comporta come lo shifter logico, ma nello spostamento a destra, gli spazi vuoti vengono riempiti con l'ultimo bit più significativo, viene utilizzato per i numeri in complemento a 2.

Shiftare a destra:

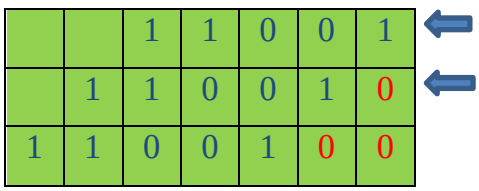
$10000 \gg> 2 = 11100$ shiftare 2 volte il valore a destra (dividere il numero per 4)



$10000 = -16 - diviso\ 4 \rightarrow 11100 = -4$ (In questo caso stiamo escludendo la parte decimale)

Shiftare a sinistra:

$11001 \ll< 2 = 00110$ shiftare 2 volte il valore a sinistra (moltiplicare il numero per 4)

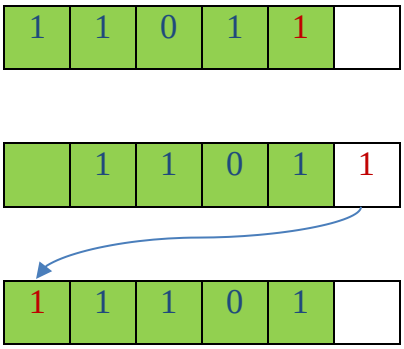


$11001 = 25 - per\ 4 \rightarrow 1100100 = 100$

Il **ruotatore** ruota un bit, shiftando un numero binario, il numero che viene tagliato fuori viene posizionato al capo opposto, essa è un operazione ricorsiva, se si esegue tante volte quanti sono i bit, si ritorna al valore originale.

$ROR =$ rotazione a destra $ROL =$ rotazione a sinistra

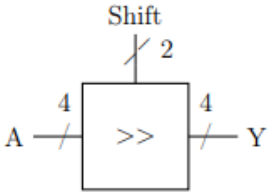
$$11011\ ROR = 11101$$



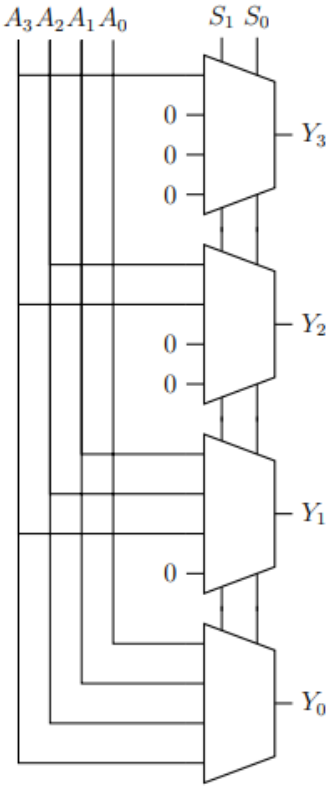
Shifter design

Il **blocco dello shifter** da N bit richiede un MUX N:1 per ogni bit, ed ci sono tanti selettori dei MUX per ogni bit da shiftare. Vediamo l'implementazione di uno **shifter logico a destra da 4 bit**:

utilizziamo 4 MUX da 2 selettori



⇒



i valori di S_1 e S_0 decreteranno quante volte i bit devono shiftare :

01 = 1 shift

10 = 2 shift

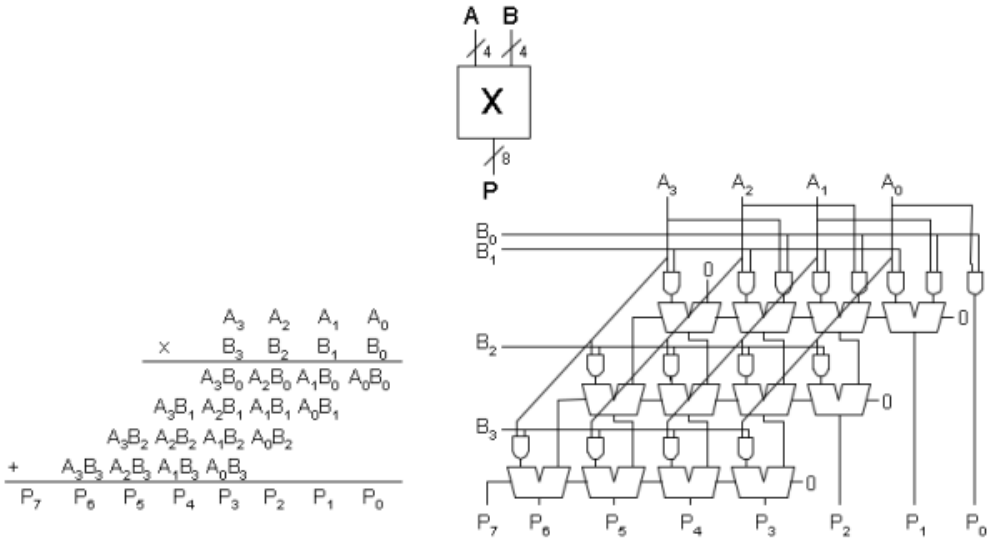
11 = 3 shift

Moltiplicatori

Le moltiplicazioni nell'ambito dei circuiti, come per il sistema binario sono una somma di prodotti parziali, ogni prodotto parziale corrisponde ad una fila di sommatori:

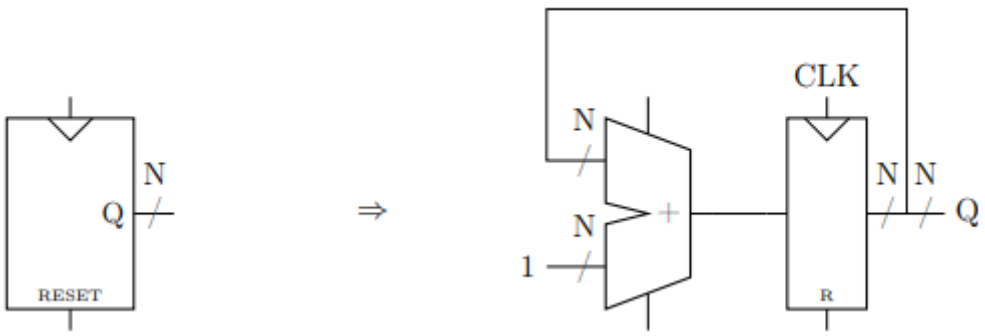
			1	0	1	x	
		1	1	1	1	=	
			1	0	1	+	
		$^{1}1$	0	1		+	⇒
	$^{1}1$	0	1			+	
$^{1}1$	0	1				=	
1	0	0	1	0	1	1	

Abbiamo bisogno di 3 file di sommatori



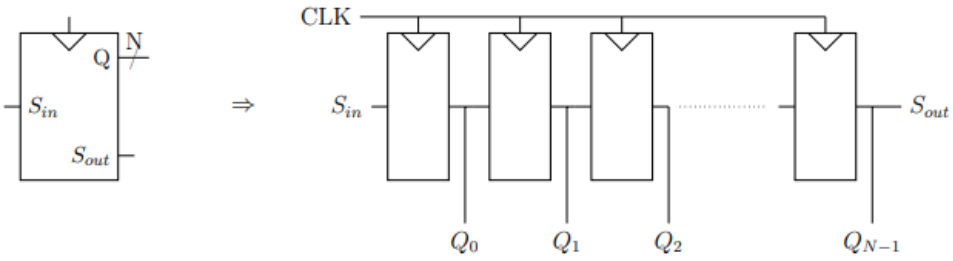
Counter

Il **contatore** incrementa il suo valore ad ogni colpo di clock, tiene conto delle istruzioni eseguite. Viene utilizzato per ciclare tra un set di numeri definiti in base alla quantità di flip flop contenuti al suo interno. (con 3 flip flop avremo la sequenza 000,001,010,011,100,101, 110,111,000,001…ecc)

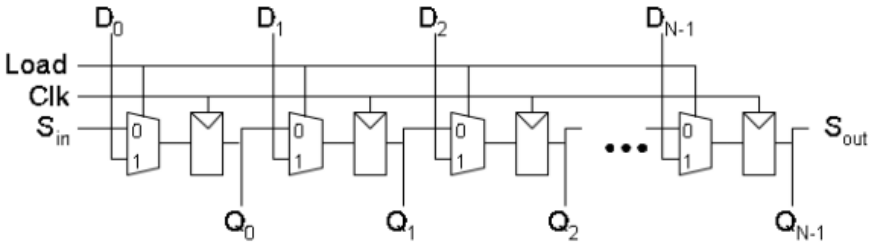


Shift register

Lo **shift register** ha un registro bit, dove ogni bit viene trasferito al registro successivo ogni colpo di clock. È connesso un output ad ogni registro, questo blocco seriale è anche chiamato serial-to-parallel converter dove l'input seriale S viene convertito in N output paralleli.



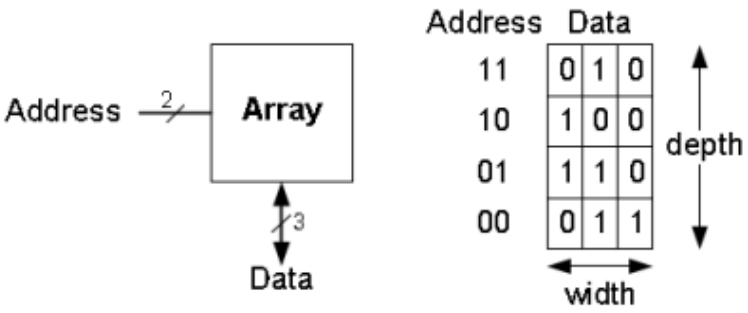
Esiste anche lo **shift register con parallel load**, dove ad ogni registro viene aggiunto un MUX che seleziona uno solo tra l'input seriale (l'output del registro precedente) ed un input esterno. Tutti i MUX vengono controllati da uno stesso segnale **Load**, Se Load = 1 allora i registri vengono caricati con il loro rispettivo "input esterno", se Load = 0 essi si comportano come un normale shift register.



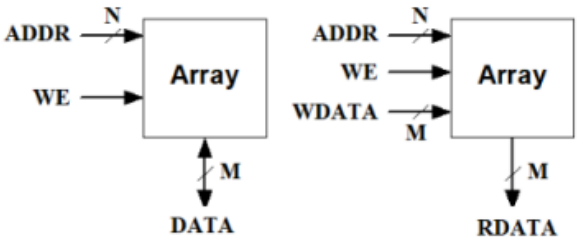
Questo blocco in base al valore del Load, può sia agire come serial-to-parallel converter, sia come parallel-to-serial converter.

Memorie

Gli **array di memoria** hanno lo scopo di immagazzinare una grande quantità di dati, ogni array di memoria possiede N input con cui si possono ottenere 2^N indirizzi univoci. Ad ogni indirizzo corrisponde una parola di M bit, in cui vengono memorizzati i dati, una memoria è quindi una matrici di 2^N righe e M colonne.



Le dimensioni della matrice sono quindi $2^N \times M$



Bisogna puntualizzare che gli M bit Data sono sia di input, sia di output, poiché possono essere utilizzati per la scrittura e per la lettura, un segnale chiamato **WE** (Write enable “abilita scrittura”) si occupa di selezionare quale delle due operazioni si intende eseguire.

Ci sono tre tipologie di memorie :

- Static Random Access Memory (SRAM)
- Dynamic Random Access Memory (DRAM)
- Read-only Memory (ROM)

Le prime due SRAM e DRAM sono definite “memorie volatili”, dato che i dati al loro interno vengono eliminati una volta scollegata la memoria dall’alimentazione elettrica, vengono quindi usate per immagazzinare dati temporanei, la ROM diversamente, è definita “memoria non volatile”, dato che i dati vengono memorizzati permanentemente, questa è però una memoria di sola lettura, non è dunque possibile sovrascrivere ciò che è memorizzato.