




 SAPIENZA

 UNIVERSITÀ DI ROMA

 DIPARTIMENTO DI INFORMATICA

Registri: caricamento/scaricamento e shifter

Prof. Daniele Gorla



 SAPIENZA

 UNIVERSITÀ DI ROMA

 DIPARTIMENTO DI INFORMATICA

Registri


Un *registro* è un insieme di n FF ordinati (per n fissato) in cui memorizzare una parola da n bit.

Le problematiche relative ai registri sono:

- caricamento e scaricamento dell'informazione
- funzionalità che fornisce il registro
- interconnessione tra registri

In base alla modalità di scrittura e lettura dei dati, abbiamo:

- Parallel Input Parallel Output (PIPO)
- Serial Input Serial Output (SISO)
- Serial Input Parallel Output (SIPO)
- Parallel Input Serial Output (PISO)



 SAPIENZA

 UNIVERSITÀ DI ROMA

 DIPARTIMENTO DI INFORMATICA

Parallel Input Parallel Output

load →

clock →

D_1 →

...

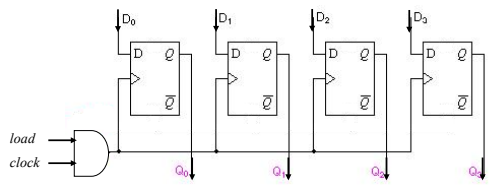
D_n →


Q_1

...

Q_n

Specifica: riceve in input n linee di dati; tali linee vengono memorizzate quando il segnale *load* è a 1 e al momento di un fronte d'onda discendente del *clock* (rete sincrona). Il valore memorizzato viene riproposto in output dopo τ .





 SAPIENZA

 UNIVERSITÀ DI ROMA

 DIPARTIMENTO DI INFORMATICA

Serial Input Serial Output (1)

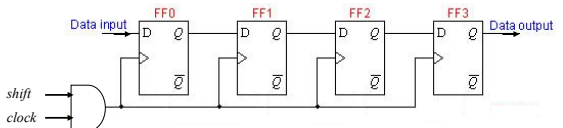
shift →


clock →

D →

→ Q

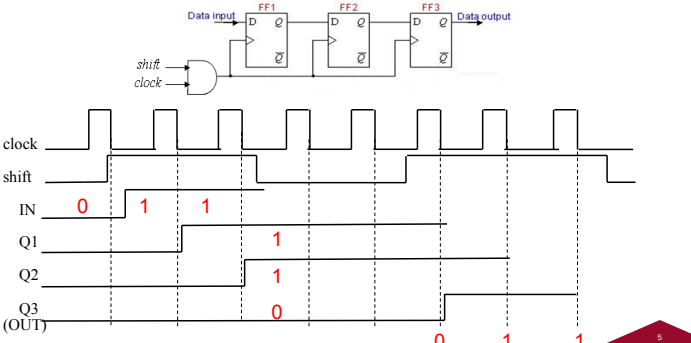
Specifica: (*caricamento*) riceve in input una linea di dati, i cui valori ad ogni fronte d'onda discendente di *clock* vengono memorizzati progressivamente, purché il segnale *shift* sia a 1. (*scaricamento*) Sull'unica linea di uscita vengono riproposti in sequenza gli n valori caricati dall'input, uno per ogni fronte d'onda discendente del *clock* in cui *shift* vale 1.




 SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA


Serial Input Serial Output (2)

Funzionamento: per caricare n bit, devo fare in modo che *shift* sia a 1 per n fronti d'onda discendenti del *clock* e che ad ogni fronte la linea dati entrante contenga il nuovo valore da memorizzare. Per scaricare il registro, bisogna settare *shift* per n cicli di *clock* e prelevare i bit dalla linea dati uscente.

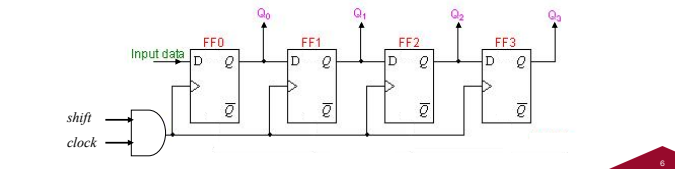



 SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Serial Input Parallel Output

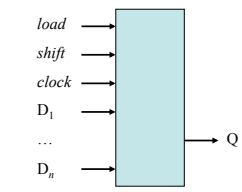


Specifica: caricamento come il SISO, scaricamento come il PIPO



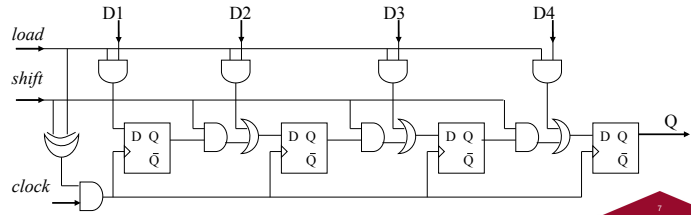
 SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA


Parallel Input Serial Output (1)



Specifica: memorizza gli n input delle linee di dati quando il segnale *load* è a 1 e al momento di un fronte d'onda discendente del *clock*. Il valore memorizzato viene scaricato quando *shift* vale 1 (in n fronti d'onda del *clock*).

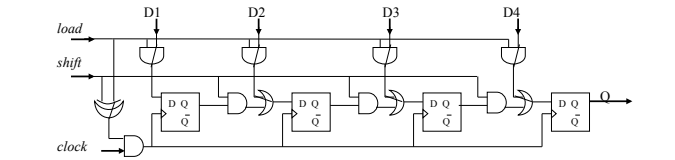
N.B.: *load* e *shift* non possono essere entrambe a 1 nello stesso istante.



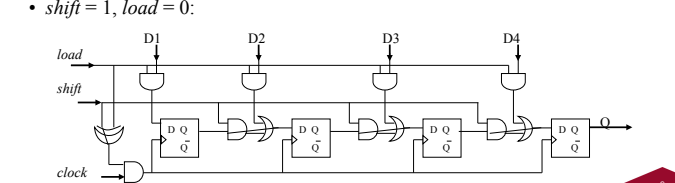
 SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Parallel Input Serial Output (2)

- $shift = load$: il loro XOR è 0, quindi il registro è in fase quiescente
- $shift = 0, load = 1$:



- $shift = 1, load = 0$:



Registri con operazioni



Indipendentemente dal tipo di caricamento/scaricamento, un registro può fornire delle funzionalità (operazioni sulla parola memorizzata).

Ci sono 2 tipi fondamentali di operazioni che un registro può fare:

- Shift
- Count

In questa lezione vedremo i vari tipi di registri shifter, nella prossima i vari tipi di registri counter.

9

Utilità dello shift



Moltiplicazione e divisione per 2 molto semplice ed efficiente:

$$00001000_2 \times 10_2 = 00010000_2$$

shift a sinistra di 1 posizione

$$00001000_2 \div 10_2 = 0000100_2$$

shift a destra di 1 posizione

OSS.1: perdo il bit più/meno significativo.

OSS.2: rimpiazzo il bit perso inserendo uno 0 all'estremità opposta.

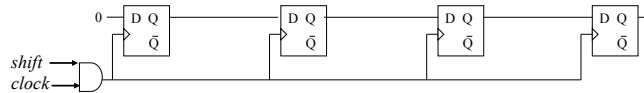
10

Shifter basilari



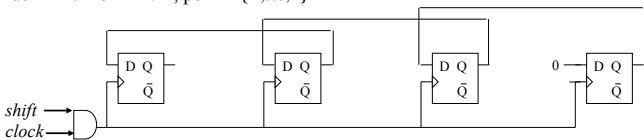
Right shifter (shift a destra):

Ad ogni fronte d'onda discendente del *clock* in cui *shift* vale 1, sposta il contenuto del FF *i* nel FF *i+1*, per $i \in \{1, \dots, n-1\}$



Left shifter (shift a sinistra):

Ad ogni fronte d'onda discendente del *clock* in cui *shift* vale 1, sposta il contenuto del FF *i* nel FF *i-1*, per $i \in \{2, \dots, n\}$



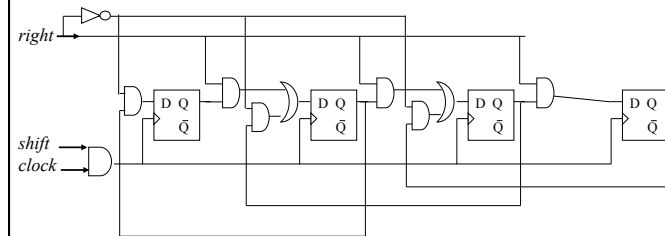
N.B.: perdita del contenuto del FF *n* (right shifter) o del primo FF (left shifter); primo FF (right shifter) e FF *n* (left shifter) riempiti con 0.

11


Shifter bidirezionale



Si comporta come un left/right-shifter in base a un bit di controllo *right*: se tale bit vale 1, fa lo shift a destra, altrimenti a sinistra:



12

 **SAPIENZA**
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Shifter con mantenimento del bit

Se lavoro con interi in complemento a 2, lo shift non funziona per fare la divisione per 2:


$$11111110_{Ca2} = -2_{10} \quad 11111111_{Ca2} = -1_{10}$$

$$11111110_{Ca2} \div 10_2 = 11111111_{Ca2}$$

Con lo shift a destra, avrei $11111110_{Ca2} \rightarrow 01111111_{Ca2} = 255_{10}!!$

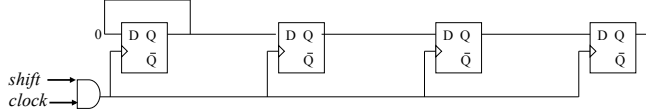
Nel fare lo shift a destra, devo quindi mantenere il bit più significativo, nel senso che non devo inserire uno 0 ma lasciare il bit che c'era.

13

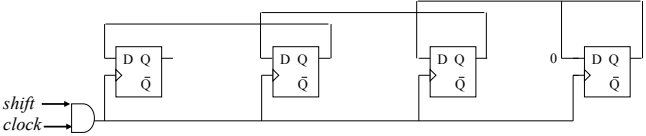
 **SAPIENZA**
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Shifter con mantenimento


Right shifter:



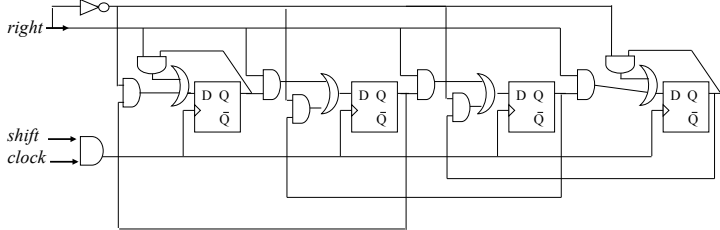
Left shifter:




14

 **SAPIENZA**
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Shifter bidirezionale con mantenimento

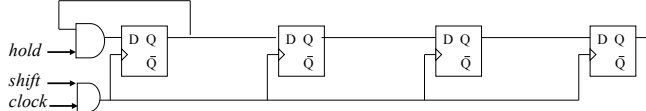


15

 **SAPIENZA**
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Shifter con/senza mantenimento

Right shifter:



In maniera analoga si può fare un left shifter e uno shifter con o senza mantenimento bidirezionale (con entrambi i segnali *hold* e *right*).

16

Registri circolari



Un altro modo per non perdere il bit che esce è reinserirlo dall'estremità opposta del registro

Si vede, cioè, il registro come chiuso circolarmente su se stesso:



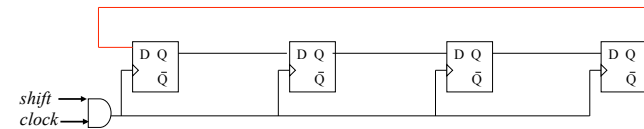
Questa operazione è molto usata quando si vogliono considerare sequenzialmente e ripetutamente tutti i bit, uno alla volta.

17

Registro circolare (antiorario)



Rotazione dell'informazione memorizzata in senso antiorario (da sinistra a destra):



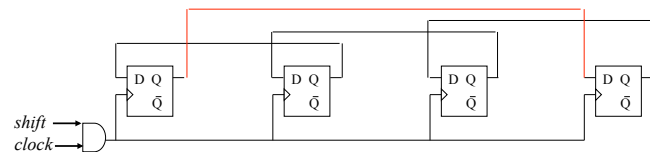
Ad ogni fronte d'onda discendente del *clock* in cui *shift* vale 1, sposta il contenuto del FF i nel FF $i+1$, per $i \in \{1, \dots, n-1\}$ e il contenuto del FF n nel primo FF.

18

Registro circolare (orario)



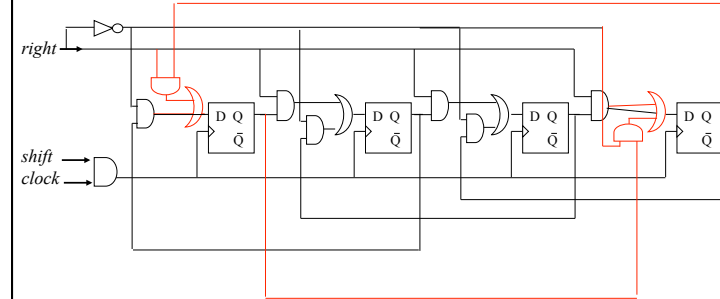
Rotazione dell'informazione memorizzata in senso orario (da destra a sinistra):



Ad ogni fronte d'onda discendente del *clock* in cui *shift* vale 1, sposta il contenuto del FF i nel FF $i-1$, per $i \in \{2, \dots, n\}$ e il contenuto del primo FF nel FF n .

19

Registro circolare bidirezionale



20