Università degli studi di Parma
Dipartimento di Ingegneria dell'Informazione

Registri e Contatori

Reti Logiche A
2016-2017

Update 14 dicembre 2016)

Docente:

prof. William FORNACIARI

william.fornaciari@elet.polimi.it
www.elet.polimi.it/~fornacia

Introduzione

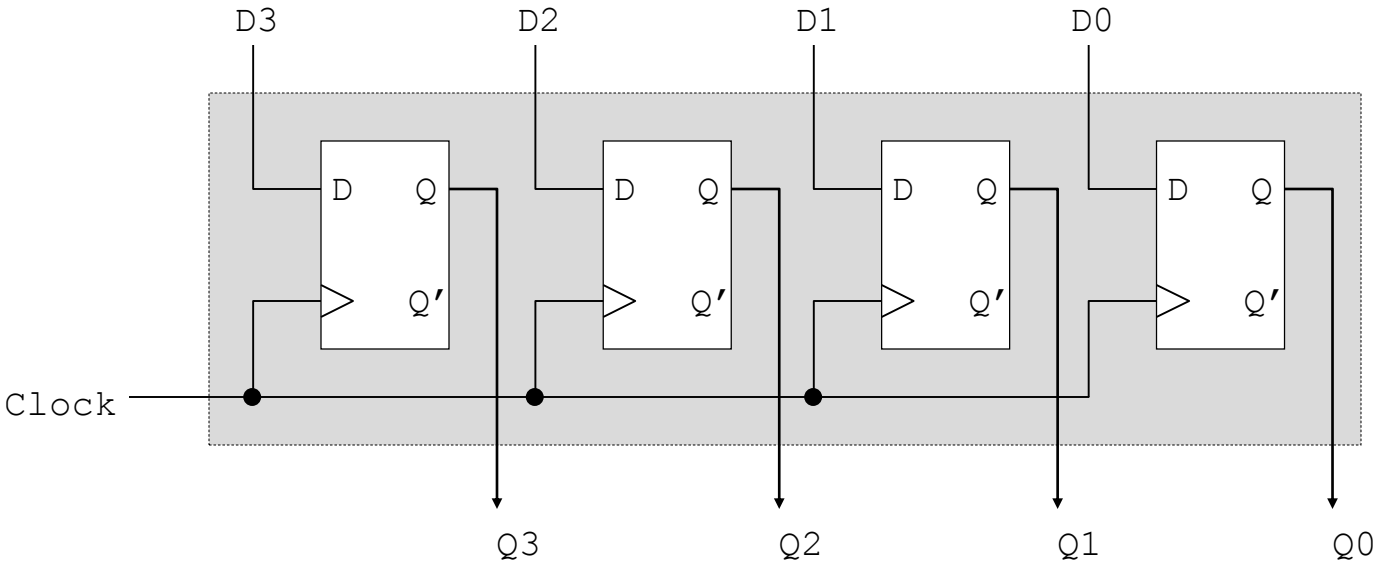
- Circuiti sequenziali speciali
  - Esiste una classe di circuiti sequenziali la cui progettazione potrebbe seguire il processo “classico” di sintesi ma che è più conveniente analizzare in altro modo
    - La regolarità della struttura facilità la progettazione
  - A questa classe appartengono:
    - Registri
      - Memorizzano una definita quantità di informazione
      - Possono operare sul contenuto una o più semplici trasformazioni.
        - » Shift destro/sinistro
        - » Caricamento parallelo/seriale
    - Contatori
      - Attraversano ripetutamente un numero definito di stati
        - » Contatori sincoroni
        - » Contatori asincorni
    - Gestori di Code
      - FIFO, LIFO

Registri

- Un *registro* è un elemento di memoria in grado di conservare un insieme di bit, denominato *parola*, su cui può eventualmente operare una o più semplici trasformazioni
  - Benché si possa utilizzare un qualunque tipo di bistabile, per realizzare i registri si preferisce utilizzare *FF D* (*master-slave* o *edge-triggered*)
- I registri si distinguono sulla base dei seguenti aspetti:
  - Modalità di caricamento dati
    - Parallelo
    - Seriale
  - Modalità di lettura dati
    - Parallelo
    - Seriale
  - Operazioni di scorrimento sui dati:
    - a destra e/o a sinistra (aritmetico o non aritmetico) e circolare

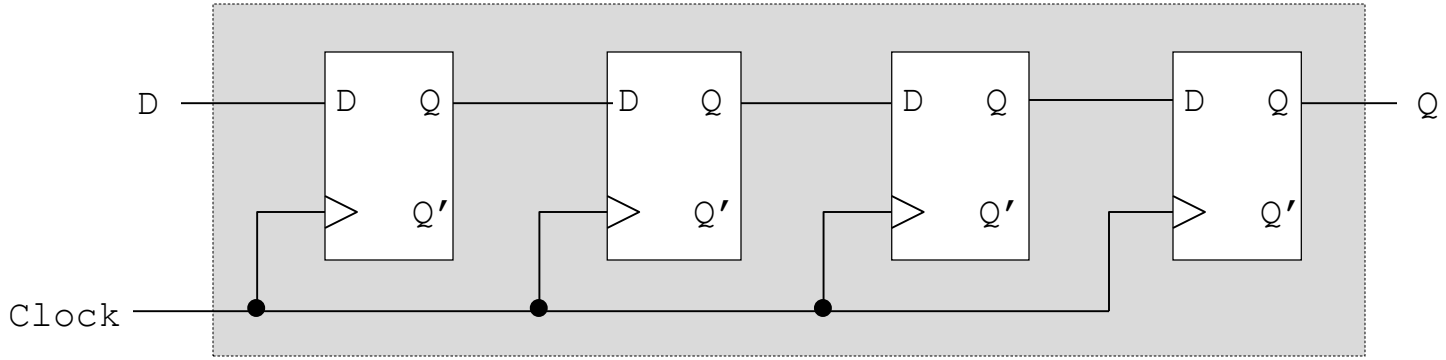
Registri

- Registro *parallelo-parallelo*
  - Esempio di registro a 4 bit



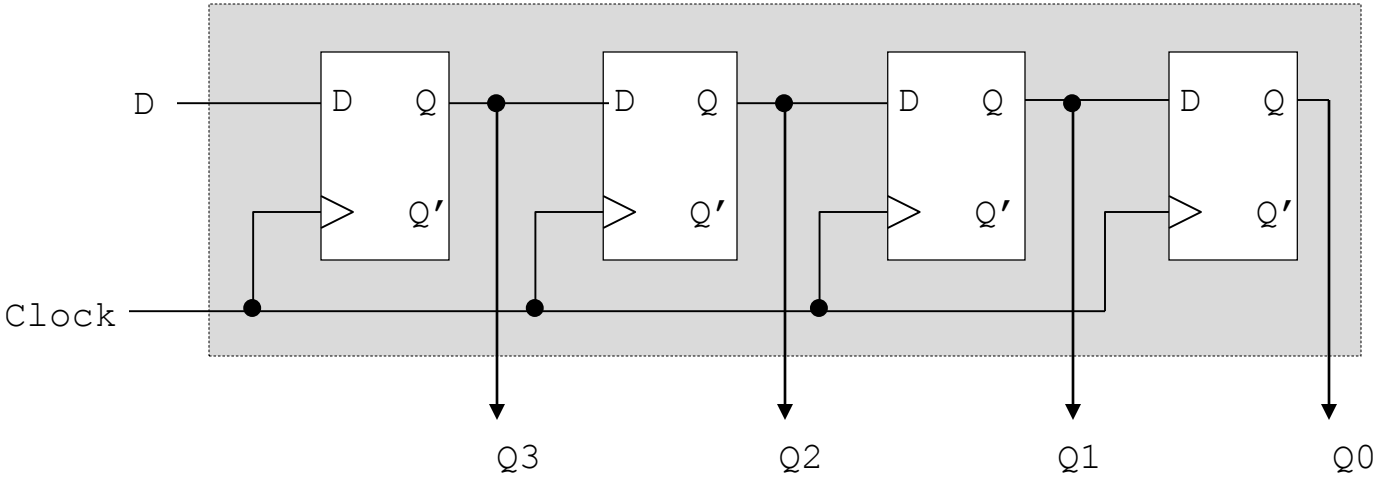
Registri

- Registro *serie-serie* (*Shift Register – Registro a Scorrimento*)
  - Esempio di registro a 4 bit



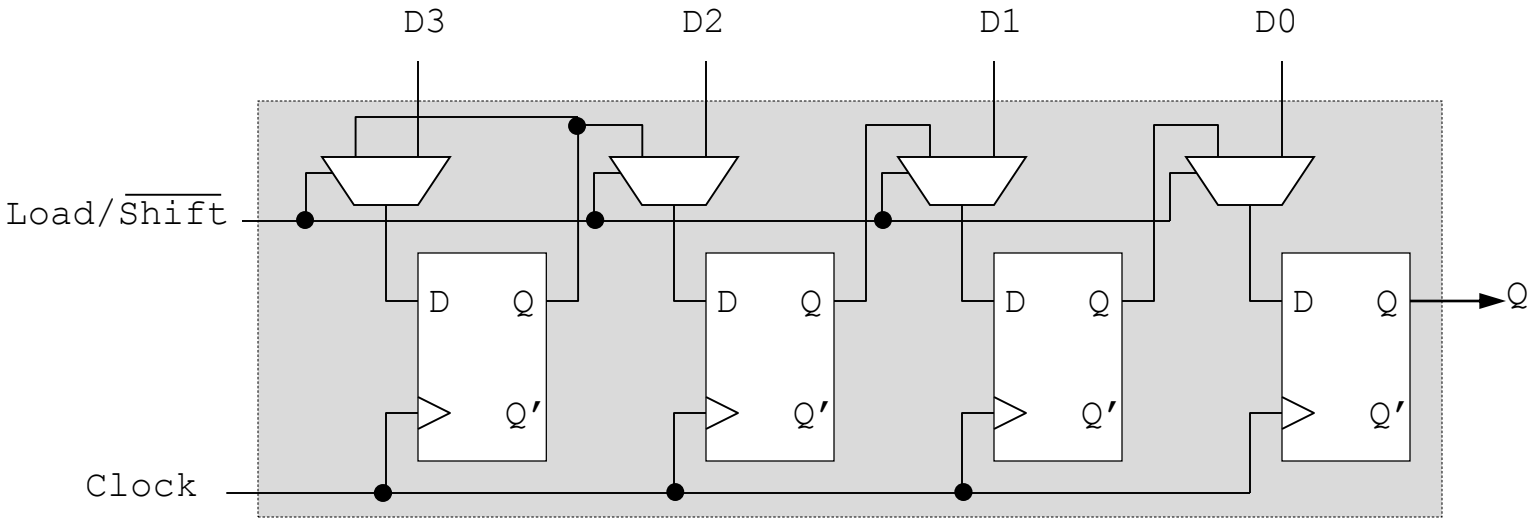
Registri

- Registro *serie-parallelo*
  - Esempio di registro a 4 bit



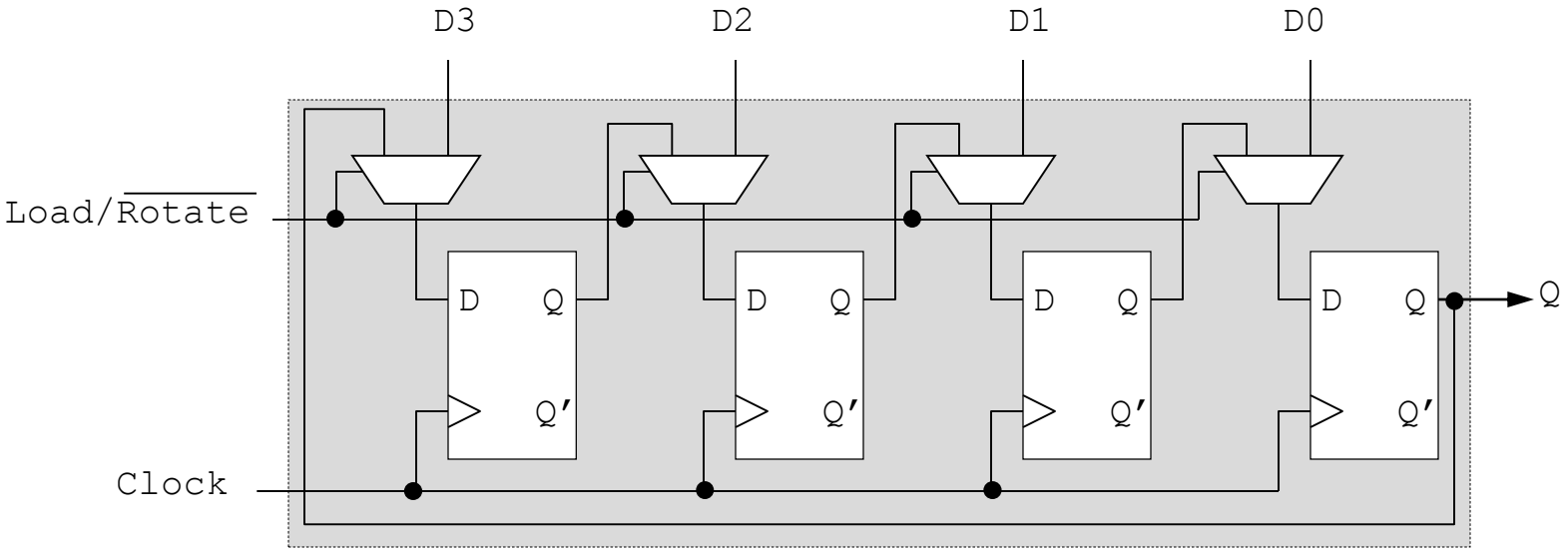
Registri

- Registro *parallelo-serie*
  - Esempio a 4 bit con shift-aritmetico (Shift Destro)
    - In fase di traslazione, ricopia il bit più significativo nella posizione più significativa (estensione del segno)



Registri

- Registro *circolare* a 4 bit
  - Esempio a 4 bit con rotazione a destra
    - In fase di traslazione, trasferisce il bit meno significativo al posto di quello più significativo, spostando i rimanenti di una posizione a destra.

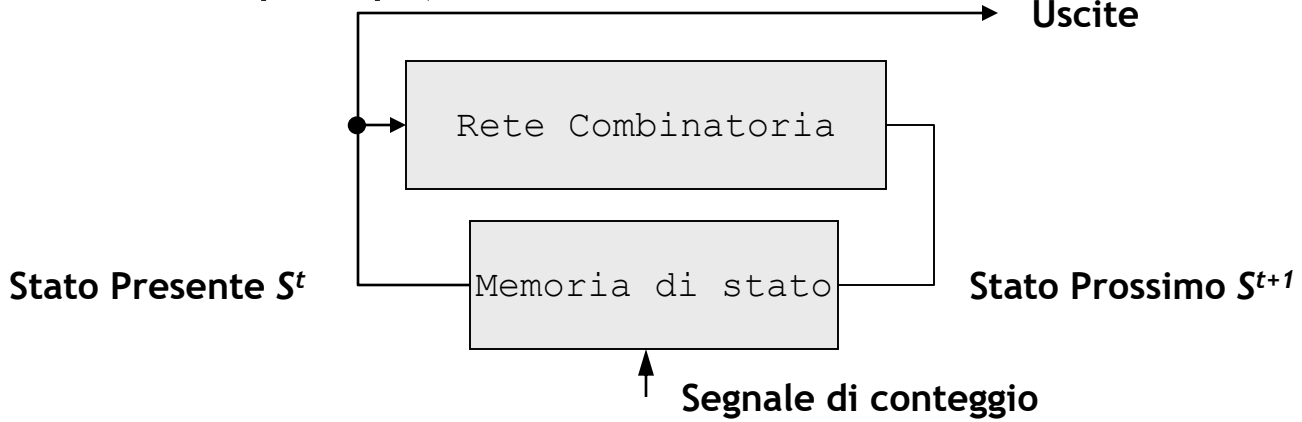


Contatori

- Un contatore è una rete *sequenziale* che, solitamente, riceve in ingresso solamente *un evento di conteggio* che sposta la posizione corrente in avanti - *upwards* – (o indietro - *downwards*) di una unità.
  - Il valore raggiunto è associato allo *stato presente*.
    - Possono esistere altri ingressi di *controllo* per la realizzazione di contatori bidirezionali; il metodo di progetto cambia di poco.
- Il contatore appartiene ad una famiglia di reti sequenziali "omogenee" caratterizzate da:
  - Specifiche di funzionamento analoghe per l'intera famiglia;
  - Ripetitività e località della struttura (in molto casi).
  - Metodologia di specifica semplificata rispetto alla generica tabella degli stati e metodologia di progetto semplificata rispetto a quella generale per le reti sequenziali;

Contatori

- Un contatore può anche essere visto come una *generica rete sequenziale dove*, a meno di particolari specifiche, *il valore d'uscita coincide con il valore di stato*.
  - A meno di casi particolari, non sono presenti reti di transcodifica
  - Si utilizzano Flip-flop *(*master slave o a commutazione sul fronte*)*



- In pratica, adottare le *tecniche generali di progetto* delle reti sequenziali risulta eccessivamente oneroso; si ricorre a tecniche *specifiche*, più semplici, secondo criteri *tipici della classe di circuiti*.
  - consentono buona *ottimizzazione*.

Contatori

- Un contatore si distingue per:
  - Il *modulo* M
    - Il contatore conterà da 0 a M-1 e, al successivo impulso, *torna a 0*;
  - Il *codice*
    - Il contatore presenta all'esterno il valore del conteggio secondo un codice stabilito.
      - A numero minimo di bit: Il numero di tali bistabili è $\lceil \log_2 M \rceil$. (es: Gray, Binario Naturale)
      - Altri codici: se il codice è su *k* bit, converrà usare *k* bistabili anche qualora fosse $k > \lceil \log_2 M \rceil$, per evitare di inserire una rete di transcodifica fra i bistabili e l'uscita del contatore. (es: 1-hot, parità)
  - La *codifica*
    - Definisce la successione degli M valori associati allo stato attraverso cui il contatore evolve.
      - Nota: la codifica dello stato è definita a priori
      - Es: M=4 - codice Gray(codice a numero minimo di bit) - codifica: S0=00 S1=01 S2=11 S3=10
      - Es: M=4 - codice Parità Pari (codice a numero non minimo di bit): codifica: S0=000 S1=011 S2=101 S3=110

Contatori

- Oltre che per *modulo*, *codice e codifica*, i contatori si distinguono in *sincroni* e *asincroni*:
  - Contatore *sincrono*:
    - Tutti i bistabili ricevono simultaneamente in ingresso l'evento di conteggio;
      - Clock oppure Gated Clock (clock attraversa una rete combinatoria).
    - Le eventuali commutazioni sono tutte simultanee (*sincrone*), a parte modeste variazioni dovute alla propagazione attraverso le reti di eccitazione dei bistabili;
  - Contatore *asincrono*:
    - Almeno un bistabile *non* riceve in ingresso il segnale di conteggio
    - La sua eventuale commutazione è comandata da quella degli altri bistabili e avverrà con un ritardo dovuto alla propagazione attraverso tali bistabili (oltre che alle reti combinatorie eventualmente presenti);
- Nel seguito si tratterà in dettaglio il progetto dei contatori sincroni

Contatori sincroni: *Contatore Binario Naturale*

- Contatore binario (modulo 2^n)
  - Modulo: 2^n; Codice: A numero minimo bit; Codifica: Binaria Naturale
  - Bistabile utilizzato: T

Tabella delle transizioni e delle eccitazioni per M= 2^1

Q_0	Q_0^*
0	1
1	0

⇒

Q_0	T_0
0	1
1	1

Tabella delle transizioni e delle eccitazioni per M= 2^2

Q_1Q_0	$Q_1^*Q_0^*$
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

⇒

Q_1Q_0	T_1T_0
0 0	0 1
0 1	1 1
1 0	0 1
1 1	1 1

Tabella delle transizioni e delle eccitazioni per M= 2^3

$Q_2Q_1Q_0$	$Q_2^*Q_1^*Q_0^*$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	1 1 1
1 1 1	0 0 0

⇒

$Q_2Q_1Q_0$	$T_2T_1T_0$
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	0 0 1
0 1 1	1 1 1
1 0 0	0 0 1
1 0 1	0 1 1
1 1 0	0 0 1
1 1 1	1 1 1

Contatori sincroni: *Contatore Binario Naturale*

- L'analisi delle tabelle delle eccitazioni evidenzia la seguente regolarità ($M=2^4$):

Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

$T_0 \equiv 1$

Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	**0**	0	0	**0**	1
0	0	0	**1**	0	0	**1**	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

$T_1 = Q_0$

Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	**0**	**0**	0	**0**	0	1
0	0	**0**	**1**	0	**0**	1	1
0	0	**1**	**0**	0	**0**	0	1
0	0	**1**	**1**	0	**1**	1	1
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

$T_2=Q_1*Q_0=Q_1*T_1$

Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	**0**	**0**	**0**	**0**	0	0	1
0	**0**	**0**	**1**	**0**	0	1	1
0	**0**	**1**	**0**	**0**	0	0	1
0	**0**	**1**	**1**	**0**	1	1	1
0	**1**	**0**	**0**	**0**	0	0	1
0	**1**	**0**	**1**	**0**	0	1	1
0	**1**	**1**	**0**	**0**	0	0	1
0	**1**	**1**	**1**	**1**	1	1	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	1
1	1	0	0	0	0	0	1
1	1	0	1	0	0	1	1
1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1

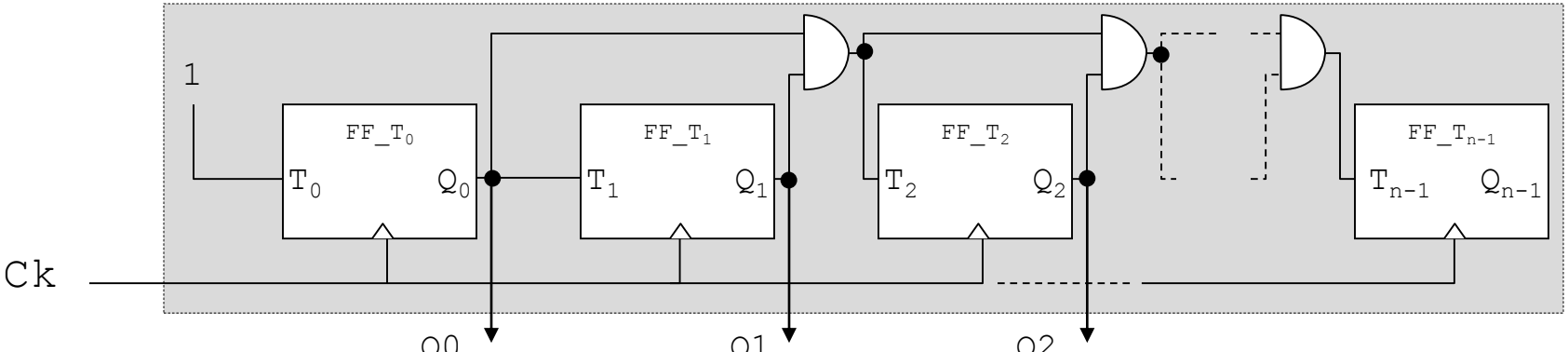
$T_3=Q_2*Q_1*Q_0=Q_2*T_2$

Contatori sincroni: *Contatore Binario Naturale*

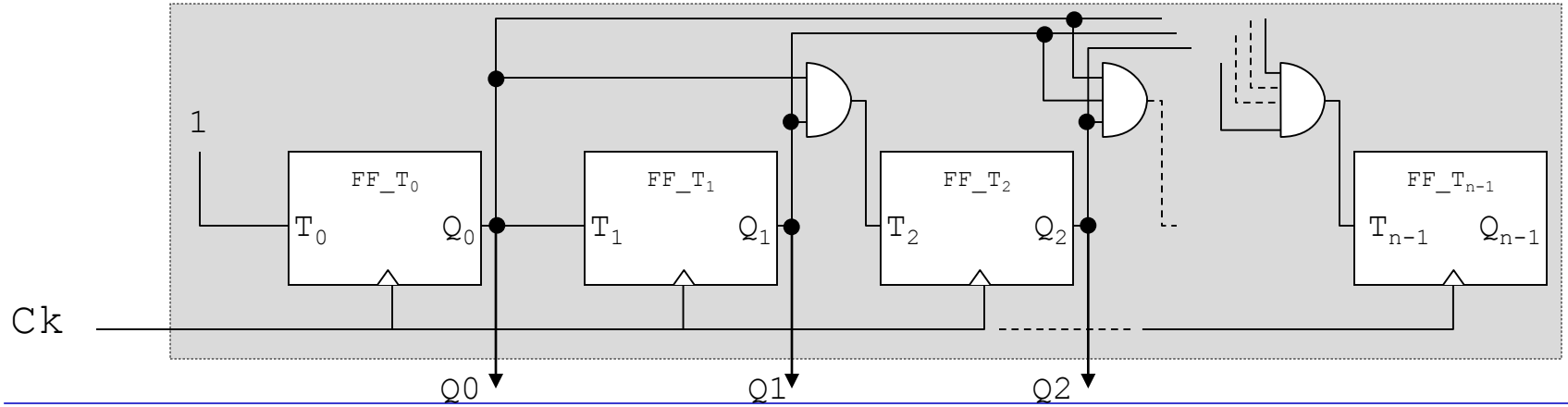
- Sono evidenti due possibili implementazioni per le funzioni di eccitazione:
  - Contatore serie: $T_0=1$; $T_1=Q_0$; $T_n=Q_{n-1}*T_{n-1}$
    - Tutti gli stadi, ad esclusione dei primi due, risultano perfettamente identici.
    - La regolarità della struttura è "pagata" con un maggior ritardo di propagazione (limita la frequenza di funzionamento).
    - Nota: la frequenza di funzionamento si riduce linearmente con la dimensione del contatore poiché T_i diventa stabile solo dopo che lo è diventato T_{i-1}.
  - Contatore parallelo: $T_0=1$; $T_1=Q_0$; $T_n=Q_{n-1}*Q_{n-2}*Q_{n-3}...*Q_0$
    - Schema molto semplice e regolare.
    - Minor ritardo di propagazione rispetto al caso precedente (frequenza di funzionamento maggiore rispetto al caso precedente).
    - Nota: la frequenza di funzionamento si riduce all'aumentare delle dimensioni del contatore a causa dell'aumento del numero degli ingressi alle porte AND.
- In generale, la regolarità deriva dal *ciclo di conteggio:* cambiando tipo di bistabile (es: FFD) le funzioni di eccitazione cambiano, ma la regolarità resta
- Un contatore binario può fungere da divisore di frequenza

Contatori sincroni: *Contatore Binario Naturale*

- Contatore binario (modulo 2^n) serie:



- Contatore binario (modulo 2^n) parallelo:



Contatori sincroni: *Contatore Binario Naturale*

- Contatore binario (modulo 2^n)
  - Modulo: 2^n; Codice: A numero minimo bit; Codifica: Binaria Naturale
  - Bistabile utilizzato: D

Tabella delle eccitazioni per M= 2^1

Q_0	D_0
0	1
1	0

Tabella delle eccitazioni per M= 2^2

Q_1	Q_0	D_1	D_0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Tabella delle eccitazioni per M= 2^3

Q_2	Q_1	Q_0	D_2	D_1	D_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

Contatori sincroni: *Contatore Binario Naturale*

- L'analisi delle tabelle delle eccitazioni evidenzia la seguente regolarità ($M=2^4$):

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	0	0	**0**	0	0	0	**1**
0	0	0	**1**	0	0	1	**0**
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	0	**0**	**0**	0	0	**0**	1
0	0	**0**	**1**	0	0	**1**	0
0	0	**1**	**0**	0	0	**1**	1
0	0	**1**	**1**	0	1	**0**	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	**0**	**0**	**0**	0	**0**	0	1
0	**0**	**0**	**1**	0	**0**	1	0
0	**0**	**1**	**0**	0	**0**	1	1
0	**0**	**1**	**1**	0	**1**	0	0
0	**1**	**0**	**0**	0	**1**	0	1
0	**1**	**0**	**1**	0	**1**	1	0
0	**1**	**1**	**0**	0	**1**	1	1
0	**1**	**1**	**1**	1	**0**	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

Q_3	Q_2	Q_1	Q_0	D_3	D_2	D_1	D_0
0	**0**	**0**	**0**	**0**	0	0	1
0	**0**	**0**	**1**	**0**	0	1	0
0	**0**	**1**	**0**	**0**	0	1	1
0	**0**	**1**	**1**	**0**	1	0	0
0	**1**	**0**	**0**	**0**	1	0	1
0	**1**	**0**	**1**	**0**	1	1	0
0	**1**	**1**	**0**	**0**	1	1	1
0	**1**	**1**	**1**	**1**	0	0	0
1	**0**	**0**	**0**	**1**	0	0	1
1	**0**	**0**	**1**	**1**	0	1	0
1	**0**	**1**	**0**	**1**	0	1	1
1	**0**	**1**	**1**	**1**	1	0	0
1	**1**	**0**	**0**	**1**	1	0	1
1	**1**	**0**	**1**	**1**	1	1	0
1	**1**	**1**	**0**	**1**	1	1	1
1	**1**	**1**	**1**	**0**	0	0	0

Parallelo: $D_0=Q_0 \oplus 1=Q_0'$ $D_1=Q_1 \oplus Q_0$ $D_2= Q_2 \oplus (Q_1 * Q_0)$ $D_3=Q_3 \oplus (Q_2 * Q_1 * Q_0)$

Serie: $D_0=Q_0 \oplus 1=Q_0'$ $D_1=Q_1 \oplus Q_0$ $D_2= Q_2 \oplus (Q_1 * Q_0) = Q_2 \oplus (Q_1 * K_0)$ $D_3=Q_3 \oplus (Q_2 * K_1)$

Contatori sincroni: *codici e moduli liberi*

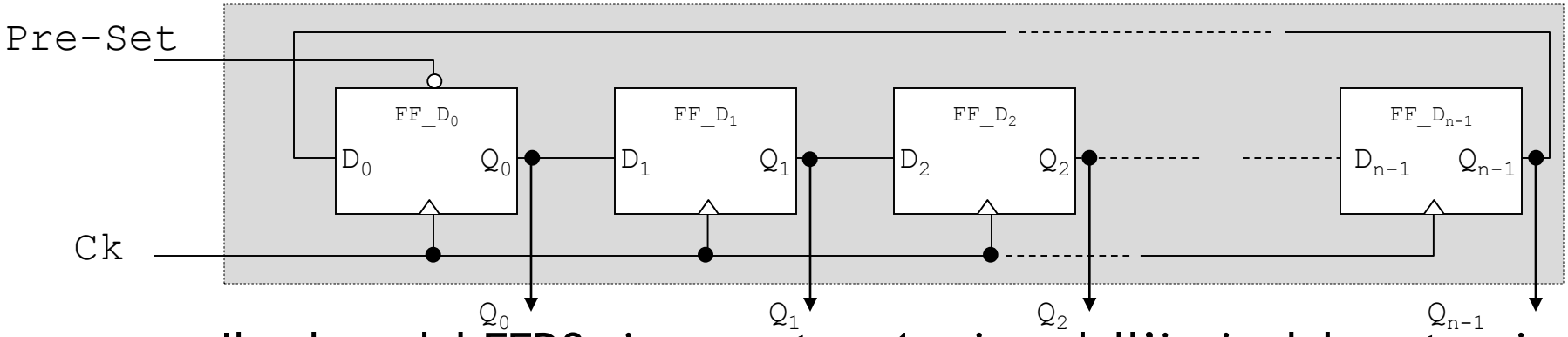
- Due casi diversi:
  - Progetto di contatori con modulo libero (2^n o diverso da 2^n), codice a numero **non** minimo bit e codifica **non** binaria naturale
    - Struttura regolare
      - Contatori ad anello (*codice one-hot*)
      - Contatore ad anello incrociato
    - A struttura non regolare
      - Si applica una metodologia di progetto semplificata rispetto a quella generale per le reti sequenziali
  - Progetto di contatori con modulo diverso da 2^n, codice a numero minimo bit e codifica binaria naturale
    - A struttura non regolare
      - Si applica una metodologia di progetto semplificata rispetto a quella generale per le reti sequenziali

Contatori sincroni: *codici e moduli liberi*

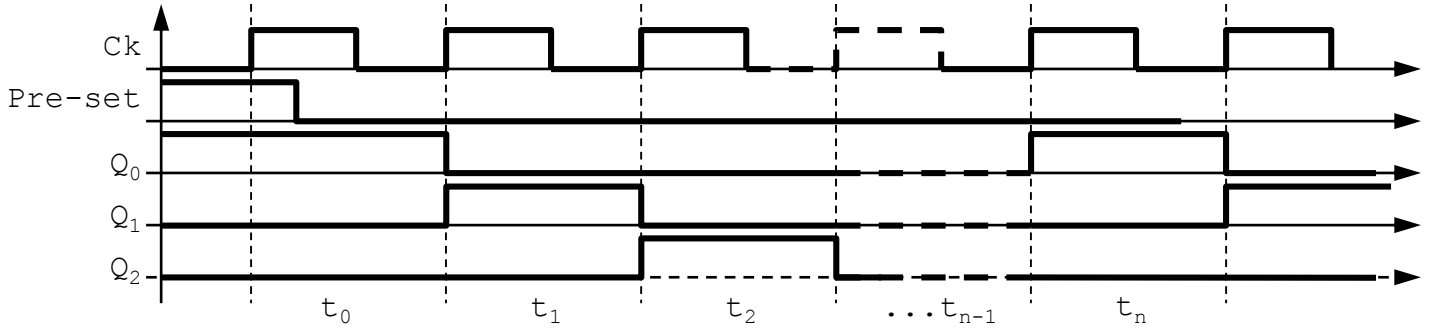
- Contatore “ad anello”
  - Modulo: n; Codice: *One hot*; Codifica: 2^k
  - Bistabile utilizzato: D
  - Codice *one-hot*:
    - In ogni codifica valida uno e un solo bit assume valore 1, tutti gli altri valgono 0
    - Per codificare *n* informazioni diverse occorrono *n* bit
      - il codice non è a numero minimo di bit
    - Esempio: i numeri da 0 a 3 sono codificati come:
      - 0 = 0001 (2^0)
      - 1 = 0010 (2^1)
      - 2 = 0100 (2^2)
      - 3 = 1000 (2^3)
      - esiste una corrispondenza 1-a-1 fra l’entità codificata e la posizione dell’unico 1 nella codifica

Contatori sincroni: *codici e moduli liberi*

- Contatore “ad anello” (*ring counter*) modulo n:
  - È un registro a scorrimento con riporto tra stadio iniziale e finale



  - Il valore del FFD0 viene posto a 1 prima dell’inzio del conteggio; i rimanenti FFD vengono posti a 0.



Contatori sincroni: *codici e moduli liberi*

- Il contatore "ad anello" ha una struttura ad alto costo ma molto semplice, compatta e veloce
  - il numero di bistabili è molto più elevato del minimo e cresce linearmente.
- Viene utilizzato in applicazioni nelle quali si deve abilitare uno e un solo sottosistema; il contatore svolge il ruolo di unità di controllo.
  - Lo stato di ogni bistabile del contatore costituisce immediatamente il segnale di controllo e non occorre alcuna rete di transcodifica.
  - Se gli stati del contatore sono n, le linee di segnale che si inviano ai sottosistemi controllati sono ancora n.
    - Nota: l'uso di un contatore con un codice a numero minimo bit – es., binario naturale – richiede una rete di *transcodifica* che per ogni stato del contatore generi un valore attivo su una sola delle n linee di segnale in uscita (rete combinatoria con $k = \lceil \log n \rceil$ ingressi e n uscite); la rete di transcodifica, al crescere di n, ha costi crescenti e introduce crescenti ritardi di propagazione

Contatori sincroni: *codici e moduli liberi*

- Contatore “ad anello incrociato” o “Johnson”
  - Modulo: 2*n (nota: sempre pari)
  - Codice e Codifica (esempio)

	Q_1Q_0	$Q_1^*Q_0^*$
0	0 0	0 1
1	0 1	1 1
2	1 1	1 0
3	1 0	0 0

	$Q_2Q_1Q_0$	$Q_2^*Q_1^*Q_0^*$
0	0 0 0	0 0 1
1	0 0 1	0 1 1
2	0 1 1	1 1 1
3	1 1 1	1 1 0
4	1 1 0	1 0 0
5	1 0 0	0 0 0

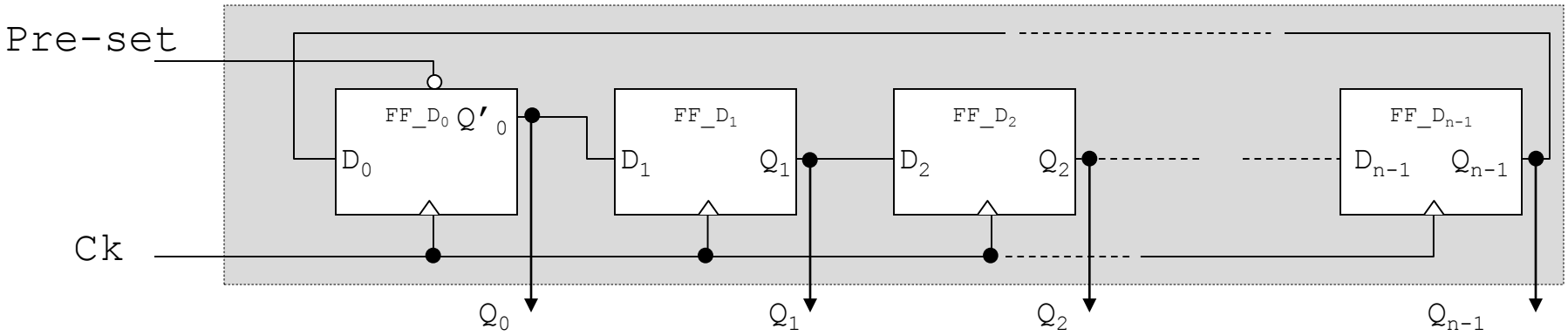
	$Q_3Q_2Q_1Q_0$	$Q_3^*Q_2^*Q_1^*Q_0^*$	Riconosce
0	0 0 0 0	0 0 0 1	Q'_3 Q'_0
1	0 0 0 1	0 0 1 1	Q'_1 Q_0
2	0 0 1 1	0 1 1 1	Q'_2 Q_1
3	0 1 1 1	1 1 1 1	Q'_3 Q_2
4	1 1 1 1	1 1 1 0	Q_3 Q_0
5	1 1 1 0	1 1 0 0	Q_1 Q'_0
6	1 1 0 0	1 0 0 0	Q_2 Q'_1
7	1 0 0 0	0 0 0 0	Q_3 Q'_2

Bistabile utilizzato: D

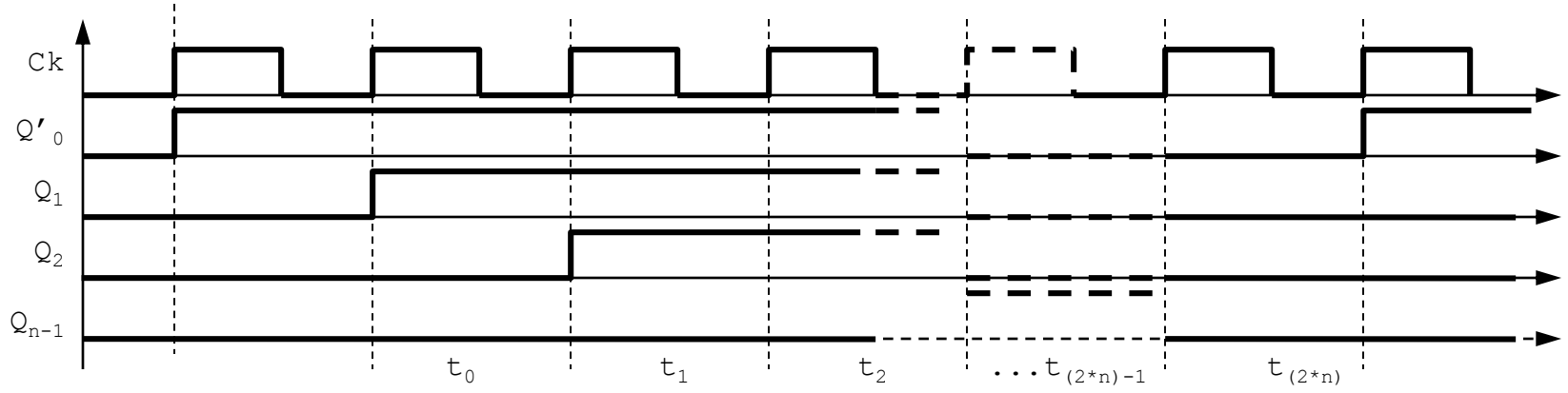
- Per codificare 2*n informazioni diverse occorrono n bit
  - Il codice non è a numero minimo di bit
- Svantaggi principali: modulo sempre pari, codice e codifica senza particolare campo di applicabilità
- Vantaggio principale: distanza di Hamming unitaria, prestazioni elevate, meno elementi di memoria rispetto al contatore ad anello
- Rete di decodifica dello stato con porte NAND a soli 2 ingressi

Contatori sincroni: *codici e moduli liberi*

- Contatore “ad anello incrociato” modulo 2*n:



  - Il valore del FFD_0 viene inizializzato a 1 (quindi Q'_0 vale 0) prima dell’inizio del conteggio; i rimanenti FFD vengono posti a 0



Contatori sincroni: *codici e moduli liberi*

- Contatore modulo diverso da 2^n: Esempio1
  - Modulo: 6; Codice: A numero minimo bit; Codifica: Binaria Naturale
  - Bistabile utilizzato: T

Tabella delle transizioni e delle eccitazioni per M= 6

$Q_2Q_1Q_0$	$Q_2^*Q_1^*Q_0^*$
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	0 0 0

⇒

$Q_2Q_1Q_0$	$T_2T_1T_0$
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	0 0 1
0 1 1	1 1 1
1 0 0	0 0 1
1 0 1	1 0 1

⇒

$T_2 = Q_1 * Q_0 + Q_2 * Q_0$

$T_1 = Q_2' * Q_0$

$T_0 = 1$

Nota: le equazioni derivano dalla sintesi delle tre funzioni combinatorie T_0, T_1 e T_2

Contatori sincroni: *codici e moduli liberi*

- Contatore modulo diverso da 2^n: Esempio2
  - Modulo: 10; Codice: A numero minimo bit; Codifica: Binaria Naturale (Contatore *BCD* o *Decadico*)
  - Bistabile utilizzato: T

Tabella delle transizioni e delle eccitazioni per M= 10

$Q_3Q_2Q_1Q_0$	$Q_3^*Q_2^*Q_1^*Q_0^*$
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 0 0 0

$Q_3Q_2Q_1Q_0$	$T_3T_2T_1T_0$
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 1
0 0 1 0	0 0 0 1
0 0 1 1	0 1 1 1
0 1 0 0	0 0 0 1
0 1 0 1	0 0 1 1
0 1 1 0	0 0 0 1
0 1 1 1	1 1 1 1
1 0 0 0	0 0 0 1
1 0 0 1	1 0 0 1

$T_3=Q_3*Q_0+Q_2*Q_1*Q_0$
$T_2=Q_1*Q_0$
$T_1=Q_3'*Q_0$
$T_0=1$

Nota: In modo analogo si potrebbe ottenere la realizzazione mediante FFD.

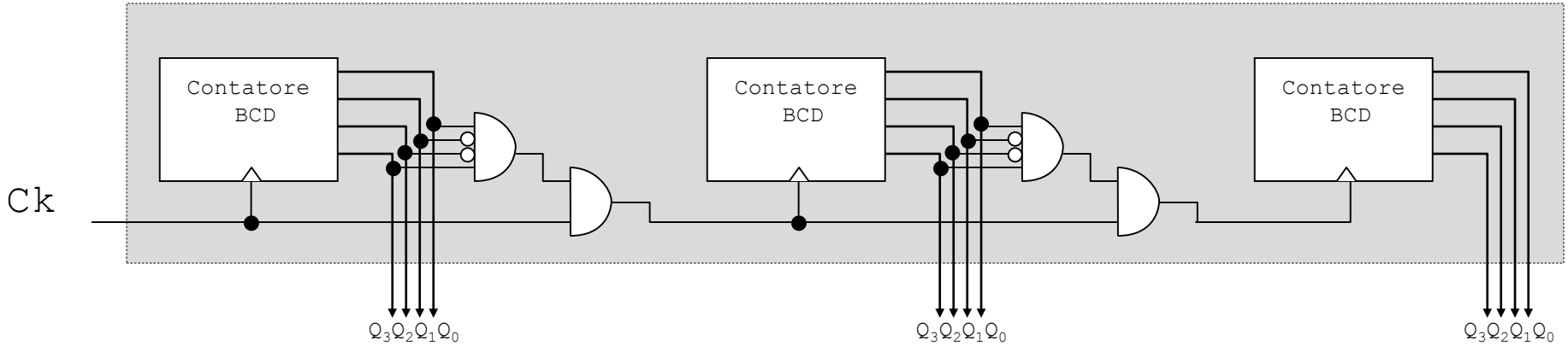
$D_0=Q_0'$
$D_1=Q_3'*Q_1'*Q_0+Q_3'*Q_1*Q_0'$
$D_2=Q_3'*Q_2'*Q_1*Q_0+Q_2*Q_0'+Q_2*Q_1'$
$D_3=Q_3*Q_1'+Q_2*Q_1*Q_0$

Contatori sincroni: *Composizione di contatori*

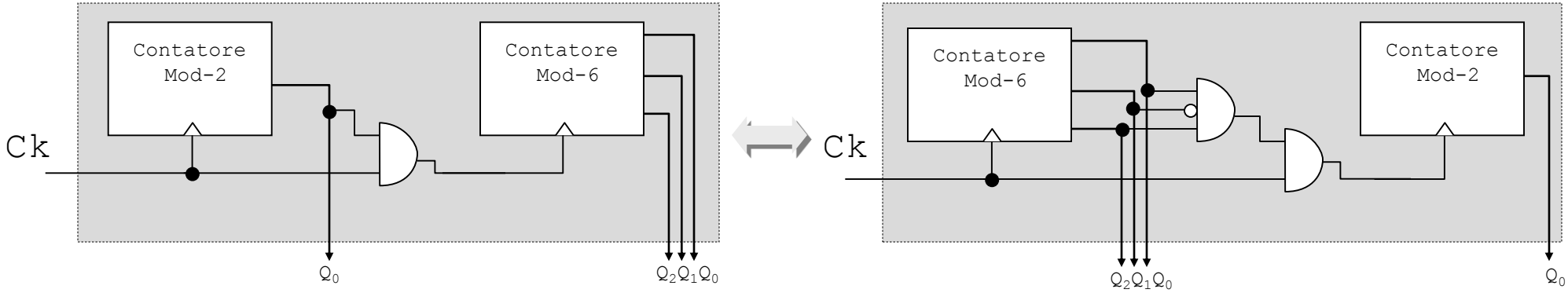
- É possibile realizzare contatori per moduli elevati partendo da contatori più semplici
  - Esempio: realizzare un contatore a k cifre decimali utilizzando K blocchi del contatore decadico (Mod-10 ([0..9]));
- Ogni sotto-contatore genera un segnale di traboccamento (*carry*) che, quando raggiunge valore 1, consente al clock di attivare il sotto-contatore collegato ad esso in cascata.
- La condizione di traboccamento è quella indicata dalla ultima configurazione di stato presente prodotta dal contatore a valle.
  - Esempio: nel contatore BCD, la condizione di traboccamento è `1001` che corrisponde a $f(Q_3,Q_2,Q_1,Q_0)=Q_3*Q_2'*Q_1'*Q_0$
- Il modulo del contatore complesso è il prodotto dei moduli.
  - Esempio: Due contatori Mod-2 e Mod-5 possono produrre un contatore decadico.

Contatori sincroni: *Composizione di contatori*

- Esempio: contatore BCD a 3 Cifre (Mod-1000)



- Esempio: contatore Mod-12 mediante composizione di un contatore Mod-2 e un contatore Mod-6 (la versione a destra è più costosa e lenta)

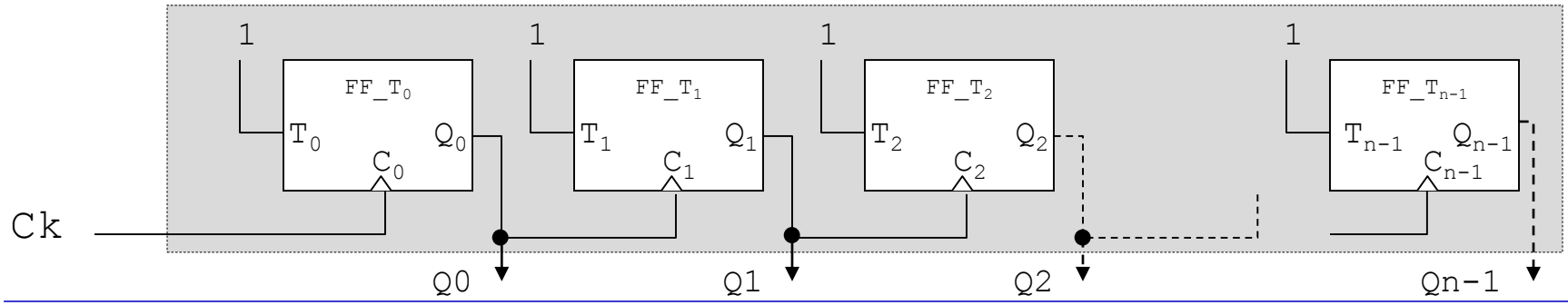


Contatori asincroni: *Generalità*

- Contatore *asincrono*:
  - Almeno un bistabile *non* riceve in ingresso il segnale di conteggio ne in modo diretto ne in modo indiretto
    - Indiretto: clock in AND con una funzione logica che ha, come supporto, alcune/tutte le variabili di stato (*gated clock*)
  - La sua commutazione dei bistabili che non sono collegati al clock è comandata da quella degli altri bistabili e avverrà con un ritardo dovuto alla propagazione attraverso tali FF (oltre che alle eventuali reti combinatorie);
  - La commutazione è generata dall'opportuno fronte di commutazione sull'uscita di almeno uno degli altri bistabili coinvolti
- *Modulo*, *codice* e *codifica* possono anche essere specificati arbitrariamente; purtrppo, può accadere che un contatore asincrono con queste caratteristiche non sia realizzabile
- Ipotesi: uso di bistabili T che commutano sul fronte ed in cui T viene posto ad 1

Contatori asincroni: *Contatore Binario Naturale*

- Contatore binario (modulo 2^n) – *Ripple Counter*
  - Modulo: 2^n; Codice: A numero minimo bit; Codifica: Binaria Naturale
  - Bistabile utilizzato: T
    - Il contatore è ottenuto collegando in cascata i bistabili
      - Con FFT che commutano sul fronte di salita il contatore *conta indietro*.
        - » *conta avanti* con FF che commutano sul fronte di discesa.
      - Funzionamento: il fronte di salita del clock modifica lo stato di FF_T_0 che passa da 0 ad 1; questo fronte di salita modifica lo stato di FF_T_1 che, a sua volta, genera un fronte di salita che modifica modifica lo stato di FF_T_2 ; ....
        - » 000...00, 111...11, 111..10, ...



Università degli studi di Parma
Dipartimento di Ingegneria dell'Informazione

Appunti da aggiungere su contatori
draft 14 dicembre 2016

Riduzione del ciclo di conteggio
Composizione di contatori
Contatori realizzando mediante shift register
Generazione di forme d'onda usando contatori J e contatori+mux
Contatori con enable
Contatori realizzati mediante ROM
Contatori up/down

Ingressi ausiliari: reset, enable, preset (load)

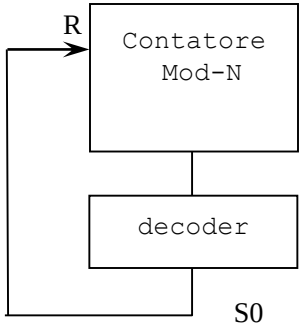
- Reset
  - Asincrono: effetto immediate di azzeramento delle uscite
  - Sincrono: azzeramento sincronizzato con il clock
- Enable
  - E=1, contatore abilitato (somma 1 ad ogni ck o evento)
  - E=0, il contatore non opera (valore congelato)
  - Utile per la composizione modulare di contatori
- Preset
  - Valore che può essere caricato nei FF interni al contatore, forza ad iniziare il conteggio da un certo stato
  - Utile per modificare il ciclo di conteggio

Modifica del ciclo di conteggio

- Estendo progettando una FSM la cui uscita diviene il pilotaggio dell'enable di un contatore
- Riduco
  - Resettando il contatore quando si raggiunge un certo valore inferiore al modulo del contatore di partenza, ovvero tagliando la coda del conteggio
  - Precaricando un valore di partenza, ovvero tagliano l'inizio del conteggio
- Potrebbe essere necessaria una rete di decodifica per fornire un nuovo significato allo stato del sistema (valori delle uscite dei FF)

Riduzione della base di conteggio

- Allo stato S0 il contatore si resetta
- Reset Sincrono
  - Raggiunto S0, R forza 0 al ciclo di clock successivo
  - La base di conteggio è S0+1
- Reset Asincrono
  - Raggiunto S0, R azzera immediatamente
  - Base di conteggio è S0
  - Rischio se un FF si resetta molto prima degli altri, di arrivare in uno stato differente dal reset globale
  - Potrebbe esserci un valore non valido per poco tempo
  - Si può mettere un D in “serie” a S0, però il ciclo è S0+2

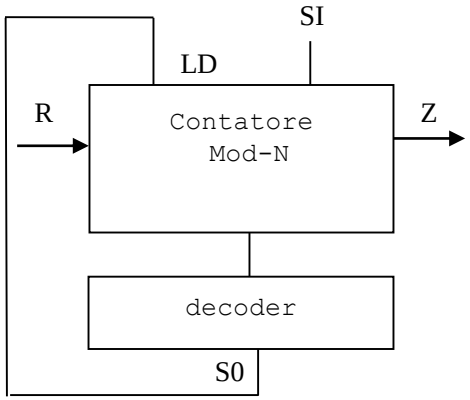


Esempio

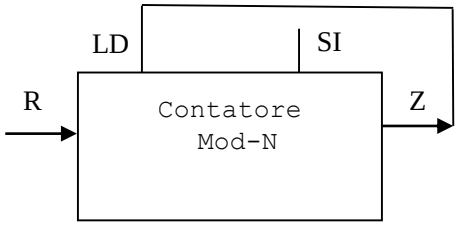
- Dato un contator per 16 ridurre la base di conteggio a 11
  - Reset sincrono
    - S0=10 (dieci), base di conteggio 10+1=11
    - Rete è un AND che riconosce 1010 ($Q_3\ Q'_2 Q_1 Q'_0$)
    - Se voglio un costo minimo posso usare anche solo $Q_3\ Q_1$

Uso del commando di caricamento LOAD

- Posso combinare il LOAD con il reset
  - Load sincrono
    - Base di conteggio S0-SI+1
  - Load asincrono
    - Base di conteggio S0-SI



- Se programmo la base di conteggio con il riporto che pilota il caricamento di un valore iniziale
  - Base conteggio = N-SI



Esempio: contatore BCD usando parallel load

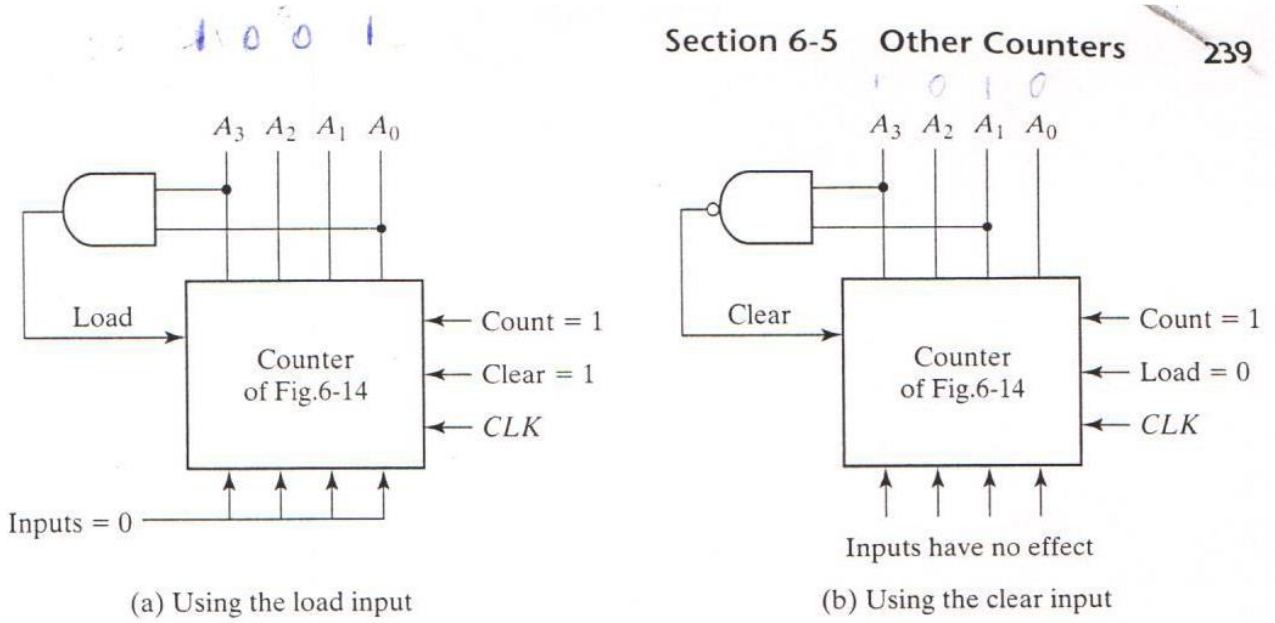
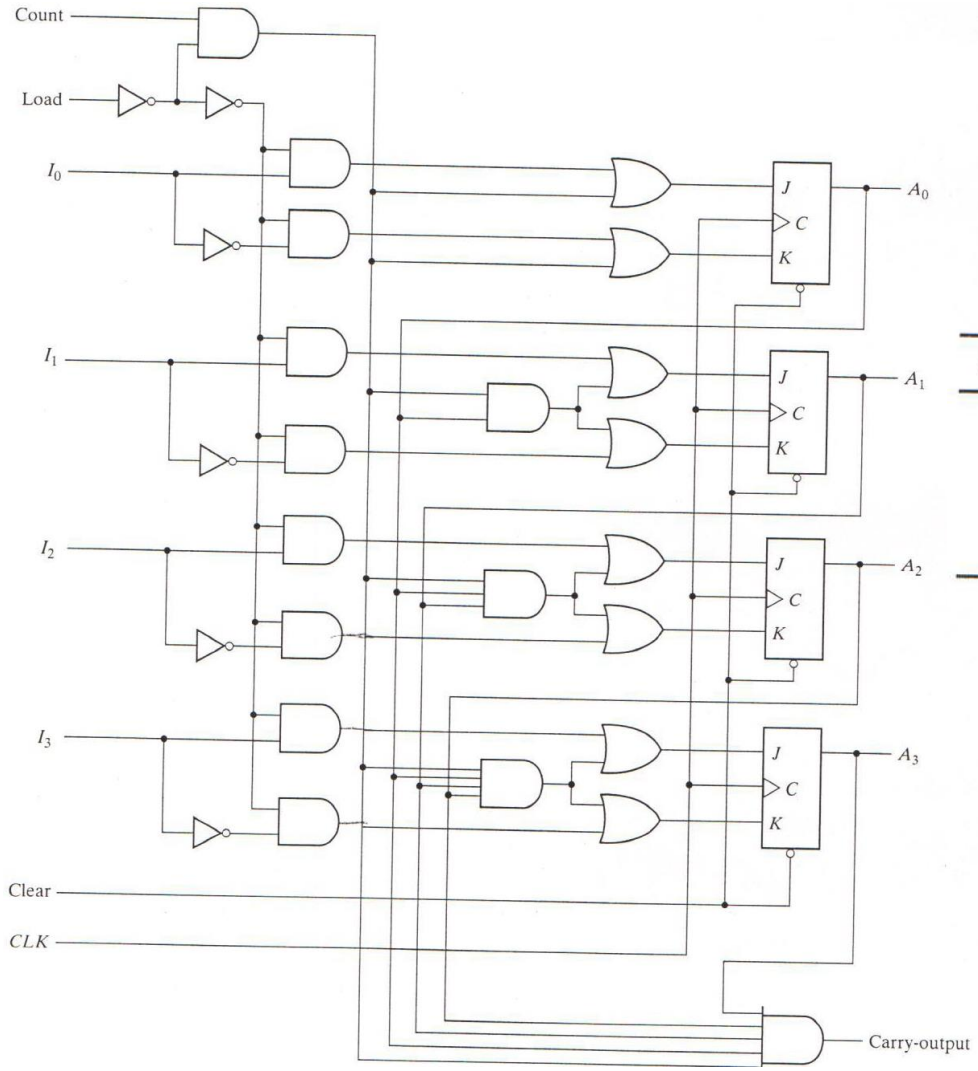


FIGURE 6-15
Two ways to Achieve a BCD Counter Using a Counter with Parallel Load

4 bit counter with parallel load and enable



Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

Contatori bidirezionali (up/down)

- Uso un segnale U, se U=1 down, se U=0 up
- Esempio con FF JK

$$J_A = K_A = \overline{U}.B.C + U.\overline{B}.\overline{C}$$

$$J_B = K_B = \overline{U}.C + U.\overline{C}$$

$$J_C = K_C = 1$$

			U=0		U=1		U=0		U=1		U=0		U=1	
A	B	C	J_A	K_A	J_A	K_A	J_B	K_B	J_B	K_B	J_C	K_C	J_C	K_C
0	0	0	0	X	1	X	0	X	1	X	1	X	1	X
0	0	1	0	X	0	X	1	X	0	X	X	1	X	1
0	1	0	0	X	0	X	X	0	X	1	1	X	1	X
0	1	1	1	X	0	X	X	1	X	0	X	1	X	1
1	0	0	X	0	X	1	0	X	1	X	1	X	1	X
1	0	1	X	0	X	0	X	1	0	X	X	1	X	1
1	1	0	X	0	X	0	X	0	X	1	1	X	1	X
1	1	1	X	1	X	0	X	1	X	0	X	1	X	1

U=0 / U=1

A \ BC	00	01	11	10
0	OX	OX	1X	OX
1	XO	XO	X1	XO

$J_A K_A (U=0)$

A \ BC	00	01	11	10
0	1X	OX	OX	OX
1	X1	XO	XO	XO

$J_A K_A (U=1)$

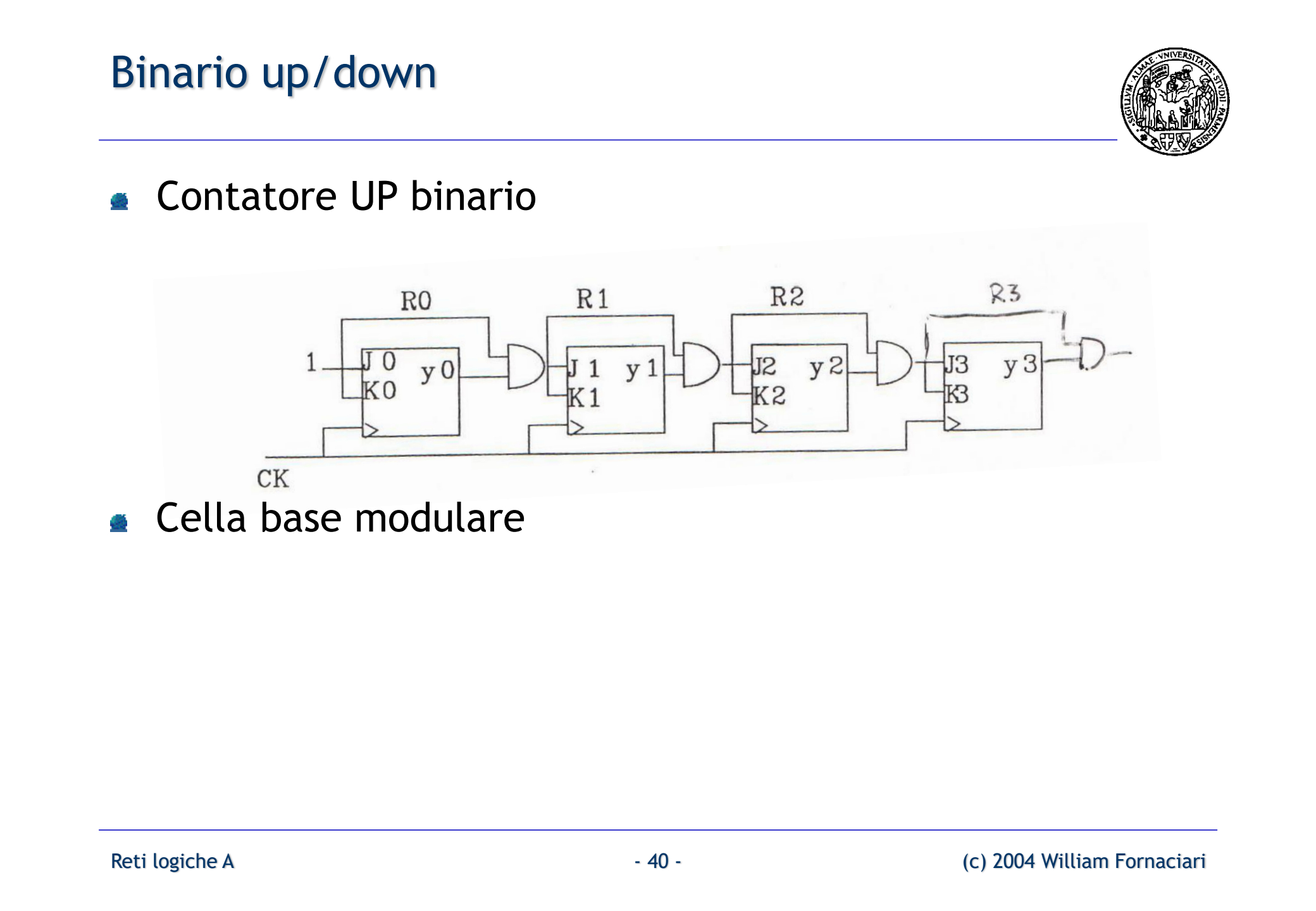
A \ BC	00	01	11	10
0	OX	OX	1X	OX
1	XO	XO	X1	XO

$J_A K_A (U=0)$

A \ BC	00	01	11	10
0	1X	OX	OX	OX
1	X1	XO	XO	XO

$J_A K_A (U=1)$

Binario up/down



- Contatore UP binario

- Cella base modulare

Contatore a ritroso

Lo stato successivo e' espresso da:

$$\frac{(y3y2y1y0)^n - 0\ 0\ 0\ 1}{(y3y2y1y0)^{n+1}}$$

$(yi)^{n+1}$ $= (yi)^n \underline{Pi}$ se non vi e' propagazione di prestito (Pi=0)

$= (\underline{yi})^n Pi$ se vi e' propagazione di prestito (Pi=1)

Quindi: $(yi)^{n+1} = (yi)^n \underline{Pi} + (\underline{yi})^n Pi = (yi)^n \text{ xor } Pi$

con: $Pi = \underline{yi-1} \cdot Pi-1$

cioe': $P0 = 1$ ... $Pi = \underline{yi-1}\ \underline{yi-2} \ldots \underline{y1}\ \underline{y0}$

Contatore binario a ritroso e up/down

La equazione caratteristica del:

FF-T

$(yi)^{n+1} = (Ti \text{ xor } yi)^n$

FF-JK

$(yi)^{n+1} = (Ji.\underline{yi} + \underline{Ki}.yi)^n$

coincide con $(yi)^{n+1}$ se poniamo:

$Ti = Pi$

$Ji = Pi \quad Ki = Pi$