-Struttura di un calcolatore:

un calcolatore è costituito dalla CPU, la main memory, la second memory e i dispositivi di I/O

-La CPU (Central Processing Unit) è l'unità centrale di elaborazione di un computer. È il componente hardware principale che esegue istruzioni di programmi, eseguendo operazioni aritmetiche, logiche, di controllo e di input/output (I/O) definite dal software. In sintesi, la CPU è il "cervello" del computer, responsabile dell'elaborazione dei dati e dell'esecuzione delle applicazioni.

-MAIN MEMORY in generale ha dimensioni limitate e accesso molto rapido. Si divide in RAM (random access memory) e ROM (read only memory); la prima è volatile, perde il suo contenuto quando si spegne il calcolatore, ed è usata per memorizzare dati e programmi; la seconda è persistente ma il suo contenuto è fisso e non può essere modificato, è usata per memorizzare programmi di sistema

-MEMORIA SECONDARIA o di massa, ha un accesso molto meno rapido, le informazioni non si perdono spegnendo il calcolatore, memorizza grandi quantità di informazioni. Esempi: dischi, CD, nastri.

-DISPOSITIVI I/O, sono usati per fare comunicare il calcolatore con l'esterno (cioè l'utente). Esempi: tastiera, mouse, video, stampante.

-BUS DI SISTEMA è la linea di comunicazione che collega tutti gli elementi di un calcolatore.

-GERARCHIA DI MEMORIA, la memoria è organizzata in modo gerarchico per consentire al microprocessore di lavorare più velocemente possibile: si parte dal microprocessore, poi ci sono le cache, la RAM, la memoria di massa (la secondaria), la memoria rimuovibile e infine i floppy disk, gli zip disk, i nastri etc..; più si scende minore sarà la velocità e maggiori le dimensioni.

-la BASE o RADICE del sistema è il numero di simboli adoperato dal sistema di numerazione (ex. Sistema decimale= 10 simboli; sistema binario= base 2= 2 simboli)

-sistema di numerazione posizionale: il valore di una cifra non dipende solo dalla cifra ma anche dalla posizione che occupa nella sequenza che rappresenta il numero

-all'interno di un elaboratore le informazioni sono rappresentate usando il sistema di numerazione binario: cioè un sistema posizionale in base 2, che utilizza solo due cifre decimali (0 e 1), che sono chiamate "bit" da "binary digit".

-con n bit si possono rappresentare 2^n valori distinti (da 0 a 2^n -1)

-un byte è costituito da 8 bit e quindi può rappresentare 256 valori distinti

-mega byte 2^20, giga 30, Tera 40

-un disco fisso da 10GB quanti bit può contenere = 10x8x2^30

-BCD (codice binario decimale), è un sistema di codifica numerica in cui ogni cifra decimale (0-9) è rappresentata da un gruppo di quattro bit. Ad esempio, il numero decimale 5 è rappresentato in BCD come 0101, e il numero 12 è rappresentato come 0001 0010.

-Il codice ASCII (American Standard Code for Information Interchange) è un sistema di codifica dei caratteri utilizzato per rappresentare testo nei computer e in altri dispositivi elettronici. Ogni carattere (lettere, numeri, simboli e controlli) è rappresentato da un valore numerico a 7 bit, che va da 0 a 127. Ad esempio, il carattere 'A' è rappresentato dal valore 65, mentre 'a' è rappresentato dal valore 97. Un carattere è un byte, un numero 4 bit.

-una parola si dirà con parità PARI se il numero degli "1" nella parola è pari; si dirà con parità DISPARI se il numero degli "1" nella parola è dispari.

-quando il calcolatore manda una parola e la legge dispari, aggiunge un "1" e diventa pari. Se arriva dispari vuol dire che la parola è arrivata con errore e da un messaggio di errore.

-codice GREY, ogni parola del codice si differenzia di un solo bit dalla precedente e dalla successiva.

-i CIRCUITI LOGICI sono componenti hardware che manipolano informazioni binarie, i circuiti base sono detti PORTE LOGICHE (logical gate). Allo scopo di descrivere i comportamenti dei circuiti digitali si può usare una algebra che specifica l'operazione di ogni porta e permette di analizzare e sintetizzare il circuito. Le variabili di questa algebra sono binarie, e possono assumere solo due valori (0,1, True o False). L'algebra dovuta a Boole è detta ALGEBRA BOOLEANA. Le variabili si indicano con le lettere A,B,C,X,Y,W,Z e le operazioni base sono AND (•), OR (+) ,NOT .

-Le funzioni booleane sono quelle funzioni di variabile booleana che possono assumere soltanto i valori vero e falso (1,0). Le funzioni booleane possono essere specificate usando la tabella di verità

-principio di Dualità: afferma che un'equazione booleana rimane valida se si considera il duale di entrambi i lati dell'uguaglianza di un'espressione. Il duale di un'espressione si ottiene: cambiando AND con OR e viceversa, cambiando uno con zero e viceversa e mantenendo le priorità implicite ed esplicite.

-Teorema di De Morgan

-Complemento di una funzione=la funzione negata. Il complemento di una funzione booleana è una nuova funzione booleana che restituisce il valore opposto della funzione originale per ogni possibile combinazione di valori di input.

-La semplificazione conduce a espressioni equivalenti, ovvero aventi la stessa tabella di verità, ma corrispondenti a implementazioni diverse.

-Teorema del consenso, data XY+ X' Z+ YZ = XY+ X'Z, con il teorema possiamo eliminare YZ poiché Y e Z appaiono in altro due prodotti che differiscono per un'unica variabile, che è diretta in un prodotto e negata nell'altro.

-Si dice forma canonica una espressione booleana che contenga una somma logica di prodotti (forma SOP, sum-of-products) ovvero un prodotto logico di somme (forma POS, product-of-sums).

-Un **mintermine** è un prodotto (AND) di tutte le variabili della funzione booleana, dove ciascuna variabile appare esattamente una volta, in forma positiva o negata. Ogni mintermine rappresenta una combinazione unica delle variabili che rende la funzione booleana uguale a 1 (vero).

-Un **maxtermine** è una somma (OR) di tutte le variabili della funzione booleana, dove ciascuna variabile appare esattamente una volta, in forma positiva o negata. Ogni maxtermine rappresenta una combinazione unica delle variabili che rende la funzione booleana uguale a 0 (falso).

-La funzione booleana può essere espressa come la somma di mintermini per cui la funzione è vera (Forma Canonica di Somma, Sum of Products - SOP). Si individuano le righe in cui F vale 1 e si scrivono i prodotti per quante sono le righe individuate; ogni prodotto è il mintermine relativo alla riga.

La funzione booleana può essere espressa come il prodotto di maxtermini per cui la funzione è falsa (Forma Canonica di Prodotto, Product of Sums - POS). Si individuano le righe in cui F vale 0 e si scrivono le somme per quante sono le righe individuate; ogni prodotto è il maxtermine relativo alla riga.

Le forme canoniche consentono di esprimere una funzione booleana con soltanto due livelli di porte logiche AND o OR.

- La mappa di Karnaugh, o K-map, è uno strumento grafico utilizzato per semplificare le funzioni booleane. È un metodo visivo che consente di minimizzare le espressioni logiche riducendo il numero di termini e la complessità delle operazioni logiche coinvolte. La K-map rappresenta le combinazioni delle variabili di input in una tabella a due dimensioni, dove ogni cella corrisponde a una combinazione specifica di valori di input e contiene il valore della funzione booleana per quella combinazione.

Caratteristiche della K-map

1. **Organizzazione**:
   - Le celle sono organizzate in modo tale che le combinazioni di input adiacenti differiscano per un solo bit (principio di Gray code).
   - Per una funzione con n variabili, la mappa di Karnaugh avrà 2^n celle.
2. **Rappresentazione**:
   - Ogni cella rappresenta un mintermine.
   - I valori della funzione (0 o 1) vengono inseriti nelle celle corrispondenti alle combinazioni di input.

- Un prodotto P è un implicante di una funzione se essa assume il valore 1 per tutti i mintermini che compongono il prodotto P considerato. Tutti gli accoppiamenti di una K-map, ottenuti accoppiando celle marcate con 1, sono implicanti. Dato un implicante P, se la rimozione di qualunque letterale dà luogo a un prodotto che non è un implicante della funzione, allora P è un primo implicante. Per una funzione F di n variabili, l'insieme dei primi implicanti corrisponde sulla mappa di Karnaugh all'insieme di tutti gli accoppiamenti costruiti utilizzando 2^m celle (m = 0, 1,…,n), contenenti 1, in cui ciascun accoppiamento dell'insieme contiene il maggior numero possibile di celle. Primo implicante essenziale: se un mintermine di una funzione è incluso solo in un primo implicante, tale termine si chiama primo implicante essenziale, cioè per una data funzione F, una cella contenente un 1, è incluso in un solo accoppiamento che è un primo implicante, allora quel primo implicante è essenziale.

-Funzioni non completamente specificate, sono funzioni la cui tabella di verità prevede, oltre alle configurazioni in cui la funzione vale 1, configurazioni per le quali sono possibili due situazioni:

Sono combinazioni di ingresso che non si possono presentare o sono combinazioni di ingresso che si possono presentare, ma per le quali è indifferente il valore che si attribuisce all'uscita. In quest'ultimo caso vengono chiamati condizioni di non specificazione e nelle k map vengono rappresentate da una X.

-Operatori funzionalmente completi: Esistono altri due operatori in grado di esprimere da soli gli altri tre: Operatore NAND, (NOT AND), simbolo (/)

Operatore NOR (NOT OR), simbolo (¯)

Una funzione booleana può quindi essere espressa: in forma SOP usando solo NAND e in forma POS usando solo NOR.

-Altri operatori sono XOR (or esclusivo) e XNOR (operatore coincidenza)

-DECODER: Un **decoder** è un circuito combinatorio che prende un input binario di n bit e attiva esattamente una delle 2^n linee di output. Ogni combinazione di input corrisponde a un'uscita unica che viene impostata su 1, mentre tutte le altre uscite sono impostate su 0. Per esempio un decoder 2-a-4 prende 2 bit di input e attiva una delle 4 uscite. Una funzione booleana di n variabili espressa in forma SOP può immediatamente essere implementata con un decoder n- 2n e una porta OR esterna ai cui ingressi si connettono le uscite del decoder corrispondenti ai mintermini della funzione.

-ENCODER: Un **encoder** è l'opposto del decoder. È un circuito combinatorio che converte una linea attiva tra molte linee di input in un codice binario sull'output. Esso ha 2^n linee di input e n linee di output. Per esempio: un encoder 4-a-2 prende 4 linee di input e le converte in 2 bit di output, indicando quale linea di input è attiva.

-MULTIPLEXER: è un dispositivo che seleziona una delle molteplici linee di input e la instrada a una singola linea di output basandosi sui segnali di selezione. La sintesi di reti combinatorie usando decoder richiede l'uso di porte OR (o NAND) esterne. Un MUX, invece, include tali porte e quindi consente una implementazione più compatta, inoltre consente di implementare una funzione booleana di n variabili, usando le n-1 variabili più significative come ingressi di selezione e la rimanente variabile meno significativa come ingresso dati, insieme alla negata, a 0 e 1.

- I circuiti sommatori, o **adders**, sono dispositivi combinatori utilizzati per eseguire operazioni di somma binaria. Esistono due tipi principali di sommatori: il **half adder** (mezzo sommatore) e il **full adder** (sommatore completo).

- Un half adder è un circuito combinatorio che somma due bit di input e produce due bit di output: la somma (S) e il riporto (C, carry).

Schema del Half Adder: Ingressi: A, B (i due bit da sommare) e Uscite: S (somma), C (riporto).

Un full adder è un circuito combinatorio che somma tre bit di input: due bit significativi (A, B) e un bit di riporto in ingresso (Cin). Il full adder produce due bit di output: la somma (S) e il riporto in uscita (Cout).

Schema del Full Adder: Ingressi: A, B, Cin (bit di riporto in ingresso) e Uscite: S (somma), Cout (riporto in uscita)

-Sommatori paralleli: circuito che produce la somma aritmetica di due numeri binari ad n-bit usando circuiti combinatori e può essere costruito a partire da n full adder (FA) connessi in cascata.

- Le reti combinatorie o circuiti combinatori sono quelli in cui l'uscita dipende solo dagli stati attuali degli ingressi. Non hanno memoria degli stati precedenti. **Esempi**:

- Sommatore
- Moltiplicatore
- Decodificatore
- Multiplexer
- Comparator

- Le reti sequenziali o circuiti sequenziali, invece, sono quelli in cui l'uscita dipende sia dagli stati attuali degli ingressi sia dagli stati precedenti. Hanno una memoria che può conservare lo stato interno.

Esempi:

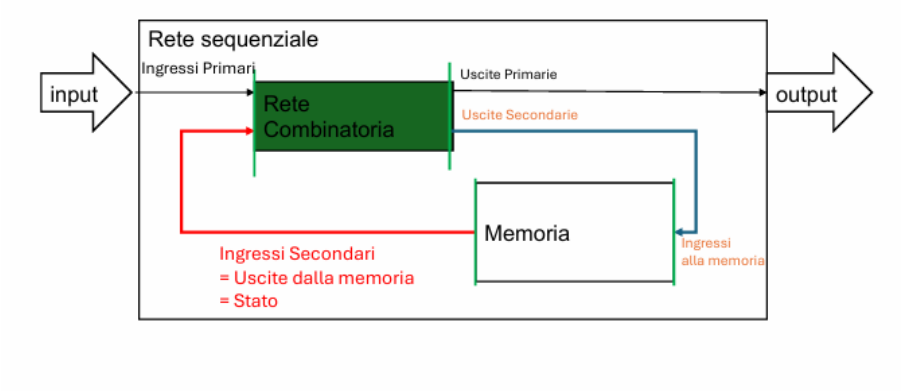
- Flip-flop (D, JK, T)
- Registri
- Contatori
- Macchine a stati finiti (FSM)
- Memorie (RAM, ROM)

- Nel modello di una rete sequenziale, è possibile distinguere: ingressi primari Ip e ingressi secondari Is, uscite primarie Up e uscite secondarie Us. Gli ingressi secondari costituiscono lo stato presente mentre le uscite secondarie costituiscono lo stato futuro.

- Si distinguono due tipologie di reti sequenziali sulla base della dipendenza tra uscite primarie e ingressi: di tipo Mealy (Le uscite primarie dipendono sia dagli ingressi primari che dagli ingressi secondari) e Moore (Le uscite primarie dipendono soltanto dagli ingressi secondari).

- I segnali che possono essere applicati ad una rete sequenziale vengono classificati in due specie:

Segnali ad impulso, si tratta di segnali la cui durata è inferiore al tempo di risposta del circuito; e segnali a livello, si tratta di segnali la cui durata è superiore al tempo di risposta del circuito.



-DIAGRAMMA DEGLI STATI: Per rappresentare una rete sequenziale è possibile impiegare tale diagramma, in cui iogni stato stabile è rappresentato da un cerchio all'interno del quale si pone la lettera che contraddistingue lo stato stesso. Supponendo che la rete si trovi nello stato A l'applicazione di una data configurazione degli ingressi può portare la rete in un altro stato (B) o farlo permanere nello stesso (A).

-reti sequenziali ASINCRONE: Le reti sequenziali le cui uscite dipendono dagli ingressi ad ogni momento, come i latch (elementi di memoria asincroni).

-reti sequenziali SINCRONE: Le reti sequenziali le cui uscite dipendono dalla conoscenza dei segnali di ingresso in dati intervalli di tempo. Una rete sequenziale sincrona impiega segnali che agiscono sulla memoria solo in istanti discreti di tempo e la sincronizzazione è ottenuta tramite dispositivi detti CLOCK generator i quali producono un treno di impulsi di cadenza.

-CLOCK: Le uscite della memoria possono cambiare i propri valori solo in presenza degli impulsi del clock. I circuiti sequenziali che usano il clock come ingresso alla memoria sono detti clocked sequential circuits. Gli elementi di memoria usati nei clocked sequential circuits sono detti flip-flop.

LATCH E FLIP FLOP: I **latch** e i **flip-flop** sono circuiti logici utilizzati per memorizzare uno stato binario. Sono fondamentali nei sistemi digitali per la costruzione di registri, contatori e altre strutture di memorizzazione. La principale differenza tra latch e flip-flop risiede nel modo in cui gestiscono i cambiamenti di stato: i latch sono sensibili al livello del segnale di controllo, mentre i flip-flop sono sensibili al fronte del segnale di clock.

Latch

Definizione

Un **latch** è un dispositivo di memoria bistabile che può essere impostato o resettato a seconda degli ingressi forniti. È sensibile al livello del segnale di controllo, il che significa che il suo stato può cambiare ogni volta che il segnale di controllo è attivo.

Tipi di Latch

1. **SR Latch (Set-Reset Latch)**:
   - Ha due ingressi, Set (S) e Reset (R), e due uscite, Q e Q' (not-Q).
   - Lo stato dell'uscita dipende dagli ingressi S e R.
   -

2. **D Latch (Data Latch)**:

- Ha un singolo ingresso, D (data), e un ingresso di controllo, Enable (E).
- L'uscita Q segue l'ingresso D quando E è attivo.

Flip-Flop

Definizione

I flip-flop sono elementi fondamentali nei circuiti sequenziali utilizzati per memorizzare un singolo bit di informazione. Sono bistabili, ovvero possono trovarsi in uno di due stati stabili, rappresentando così un bit (0 o 1). I flip-flop sono componenti importanti nella costruzione di registri, contatori e memorie digitali. Un **flip-flop** è un dispositivo simile a un latch, ma è sensibile al fronte del segnale di clock (positivo o negativo). Ciò significa che il flip-flop può cambiare stato solo in risposta a un cambiamento nel segnale di clock, rendendolo adatto per applicazioni sincrone.

Tipi di Flip-Flop

1. **SR Flip-Flop**:
   - Ha due ingressi, S e R, e un ingresso di clock (CLK)
2. **D Flip-Flop**:

- Ha un singolo ingresso D e un ingresso di clock (CLK).

3. **JK Flip-Flop**:

- Simile al SR Flip-Flop, ma con gli ingressi J e K che permettono di evitare l'indeterminatezza quando S=R=1.

4. **T Flip-Flop**:

   Ha un singolo ingresso T (toggle) e un ingresso di clock (CLK).

Costruzione di un Flip-Flop da Latch

I flip-flop possono essere costruiti combinando due latch in modo tale da creare una sensibilità ai bordi del clock.

- **Master-Slave D Flip-Flop**:
  - Costruito utilizzando due latch D in cascata. Il primo latch (master) è attivo quando il clock è basso, e il secondo latch (slave) è attivo quando il clock è alto. Questo schema garantisce che il flip-flop cambi stato solo sui bordi del clock.
- **Edge-triggered:**

  i latch sono connessi in modo che il flip flop ottenuto commuti solo in presenza del fronte di salita (o di discesa) del clock, ovvero della sua transizione da 0 a 1 (o da 1 a 0). Si dice positive edge-triggered flip flop, se attivo sul fronte di salita, mentre si dice negative edge-triggered flip flop, se attivo sul fronte di disces

ANALISI DELLE RETI SEQUENZIALI:

Può essere effettuata sfruttando quelle che vengono definite equazioni caratteristiche (oppure equazioni di eccitazione) dei diversi FF. Queste equazioni forniscono una espressione analitica che lega l'uscita del
FF all'istante n+1 con gli ingressi e l'uscita all'istante n. A tal fine si può utilizzare una mappa simile a quella di Karnaugh, nella quale è insito il concetto di tempo, nota come mappa delle transizioni (oppure mappa composita).

Procedura di analisi:

1. Etichettare le uscite dei FF presenti nella rete
2. Scrivere le equazioni delle uscite secondarie (equazioni di ingresso dei FF) e delle uscite
   Primarie
3. Sostituire le equazioni di ingresso FF nelle equazioni caratteristiche, in modo da ottenere le
   equazioni dello stato futuro
4. Costruire una tabella Stato Presente/Stato Futuro utilizzando le equazioni caratteristiche dei FF e le equazioni di uscita ottenute ai passi 2 e 3
5. Illustrare il comportamento della rete tramite un diagramma a stati (automa a stati finiti) o diagramma di temporizzazione a partire dalla tabella Stato Presente/Stato Futuro

SINTESI DELLE RETI SEQUENZIALI SINCRONE:

La sintesi di una rete sequenziale inizia, così come quella di una combinatoria, con una descrizione verbale, dalla quale ottenere la tabella di stato o l'equivalente diagramma di stato. • Una rete sequenziale sincrona è costruita tramite l'uso di una rete combinatoria e flip flop, il numero dei flip-flop è determinato dal numero degli stati della rete; n flip-flop permettono di rappresentare 2n stati.
SEQUENCE RECOGNIZER: Il circuito che vogliamo sintetizzare deve riconoscere occorrenze di una particolare sequenza di bit all'interno di sequenze più lunghe. Il fattore chiave è che la rete deve avere memoria del passato, cioè ricordare i valori precedenti dell'ingresso; per esempio per la sequenza 1101 il circuito deve essere in uno stato che deve "ricordare" che i precedenti tre ingressi erano 110. Possiamo rappresentare la rete usando un diagramma a stati, ogni stato lo possiamo etichettare con una lettera dell'alfabeto.

TABELLE DI PILOTAGGIO/ECCITAZIONE: Durante la progettazione di una rete sequenziale, sono note le transizioni dallo stato presente state allo stato futuro; Si desidera trovare le condizioni di ingresso del flip-flop che causano tali transizioni. Occorre, allora, una tabella che elenca gli ingressi necessari per ottenere un cambio di stato, ovvero una tabella di eccitazione/pilotaggio. Nelle colonne che rappresentano gli ingressi al FF, viene riportato il valore che deve avere l'ingresso per avere la transizione dallo stato presente allo stato futuro.

RIDUZIONE DEGLI STATI: La riduzione del numero di FF in una rete sequenziale (RS) prende il nome di riduzione degli stati. Ma spesso, una riduzione di stati comporta un incremento di complessità della rete combinatoria (RC). La riduzione degli stati si basa sul concetto di stati equivalenti;

STATI EQUIVALENTI: Due stati si dicono equivalenti se per ciascuna variabile di ingresso essi producono esattamente le stesse uscite ed evolvono nello stesso stato o in stati equivalenti. In altre parole, perché due stati siano equivalenti occorre verificare che, per ciascuna configurazione degli ingressi, si produca la stessa uscita e si arrivi al medesimo stato o a stati equivalenti. Quando due stati sono equivalenti, è possibile rimuovere uno dei due senza alterare il comportamento esterno della rete sequenziale.

Tabella di implicazione: In alcuni casi, l'equivalenza tra due stati dipende dalla eventuale equivalenza di un'altra coppia di stati, come nel caso di d ed f. La tabella di implicazione estende il metodo di riduzione visto, esplorando sistematicamente tutte le possibili equivalenze. Si costruisce una tabella triangolare di tutte le possibili coppie di stati, prese una sola volta. Ciascuna cella conterrà:

• X se gli stati non sono equivalenti

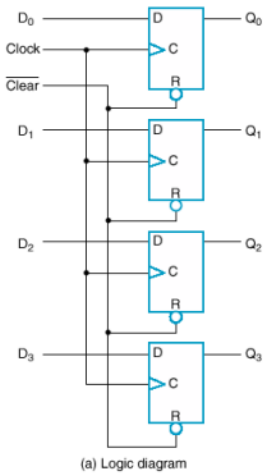
• ~ se i due stati sono equivalenti

• L'elenco di stati da cui eventualmente dipende l'equivalenza

REGISTRI E CONTATORI:

Registri e contatori sono *blocchi funzionali sequenziali* usati per la progettazione di sistemi digitali.
• I REGISTRI sono utili per la memorizzazione e la manipolazione delle informazioni.
• I CONTATORI servono per la sequenzializzazione e il controllo delle operazioni in un sistema digitale.

-Un registro include un insieme di FF. Se un FF ha la capacità di memorizzare un bit di informazione, un registro ad n-bit, che include n-FF, è capace di memorizzare n bit di informazione binaria. Il termine *registro* è applicato a un insieme di FF, possibilmente insieme a porte logiche, che esegue l'attività di elaborazione di dati. I FF memorizzano i dati; le porte logiche determinano il nuovo dato che deve essere trasferito nei FF. Il più semplice dei registri consiste di soli FF senza gate esterni.
Il Clock comune in ingresso ai 4 FF fa commutare tutti i FF sul fronte di ogni impulso di clock.
I dati binari disponibili ai 4 ingressi D sono trasferiti nello stato Q a 4-bit del registro.
L'ingresso R' ai FF, è collegato al segnale *Clear* perché è attivo con segnale 0. Questo segnale è asincrono.



(a) Logic diagram

Clear si mantiene ad 1 durante le normali operazioni, in tal modo si ha il livello zero solo quando è desiderato il reset (diversamente porterebbe a ritardi e malfunzionamenti). Il trasferimento di una nuova informazione dentro un registro è nota come *caricamento (loading)*. Se tutti i bit dei registri sono caricati simultaneamente, ad un comune impulso di clock, si parla di **caricamento parallelo**.

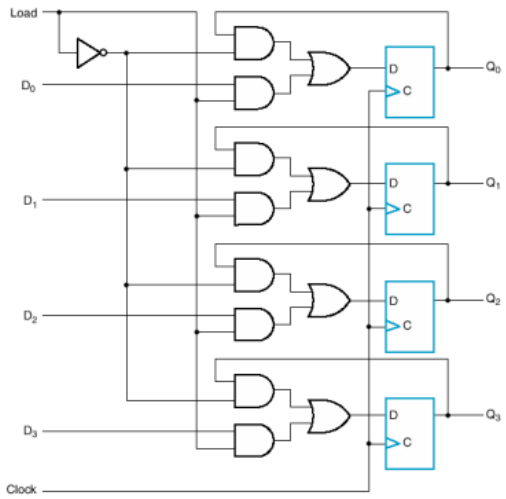


Fig. 5-2 4-Bit Register with Parallel Load

- L'ingresso **Load** stabilisce l'azione che deve essre presa in presenza dell'impulso di clock.
- Quando **Load = 1** i dati dai quattro ingressi sono trasferiti nei FF in presenza dell'impulso di clock.
- Quando **Load = 0** i dati di ingresso sono bloccati e i D input dei FF sono connessi alle uscite.
- La retroazione è necessaria affinchè sia rispettata la condizione di "no change". Per lasciare le uscite invariate è necessario che il D-input sia uguale al valore presente dell'uscita.
- E' da notare che gli impulsi di clock sono continuamente applicati agli ingressi C di controllo. **Load** determina se il prossimo impulso deve accettare la nuova informazione presente agli ingressi. L'informazione ai 4 FF è trasferita simultaneamente e questa tecnica è preferibile al clock-gating perché evita il clock-skew.

LOAD CONTROL INPUT: Molti sistemi digitali hanno un *master clock generator* che fornisce una serie continua di impulsi di clock. Il master clock si comporta come il cuore che fornisce un battito costante a tutte le parti del sistema.
Se il contenuto dei registri non deve essere cambiato, il clock deve essere ostacolato nel raggiungimento dell'ingresso di controllo dei FF.
Per governare quei cicli di clock durante i quali l'impulso di clock non deve avere effetto sui registri, si usa un segnale di controllo separato, cioè il *load control input* (*Load*), combinato con il clock come di seguito mostrato:

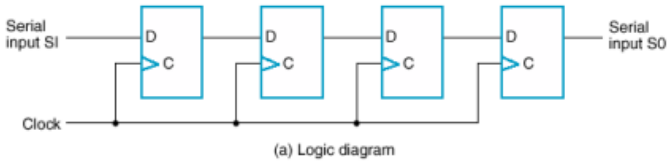
$$\text{C inputs} = \overline{\text{Load}} + \text{Clock}$$

Quando Load = 1 allora C inputs = Clock e il registro riceve il clock normalmente, la nuova informazione può essere trasferita dentro il registro su una transizione positiva.

Quando Load = 0, C inputs = 1 non si hanno transizioni e i nuovi dati non possono essere caricati nel registro. Per un corretto funzionamento Load signal deve mantenere il valore costante 0 oppure 1 anche quando il clock=0.

CLOCK GATING: il clock è connesso al C input del registro tramite l'uso di porte logiche. Inserendo dei gate lungo il percorso del clock si producono differenti ritardi di propagazione tra Clock e gli ingressi dei FF aventi o non aventi clock-gating. Se il segnale di clock arriva ai FF o registri in tempi differenti si dice che esiste un ***clock skew***. Affinché si abbia un vero sistema sincronizzato il clock deve arrivare simultaneamente a tutti i FF, per tal ragione è consigliabile che le operazioni di controllo del registro avvengano senza l'uso del clock-gating se possibile, altrimenti bisogna adoperarsi in modo da rendere il clock skew il più possibile vicino a zero.

REGISTRO A SCORRIMENTO: o shift register, è un registro capace di spostare i suoi bit memorizzati in una o entrambe le direzioni. La sua configurazione logica consiste in una catena di flip-flop in cascata, con l'uscita di un FF connessa all'ingresso del prossimo. Tutti i FF ricevono un comune impulso di clock il quale attiva lo shift da ogni posizione alla prossima.



(a) Logic diagram

-Un *contatore* è un registro che si muove attraverso una predeterminata sequenza di stati a causa dell'applicazione dell'impulso di clock. Le porte del contatore sono connesse in modo da produrre la prescritta sequenza di stati. I contatori sono disponibili in due categorie:
ripple counter e **synchronous counter**.

In un ripple counter le uscite di un FF servono per il *triggering* di altri FF. In altre parole l'ingresso di controllo *C* di alcuni o tutti i FF non è sensibile al clock ma alle transizioni che occorrono alle uscite di altri FF.

Contatori a cascata

- Il diagramma logico di un 4-bit ripple counter è mostrato in figura. Il contatore è costruito usando FF capaci di complementare il loro contenuto: JKFF.
- L'uscita di ogni FF è connesso al C input del successivo FF.
- Il FF che mantiene il bit meno significativo è sensibile al clock.
- La commutazione del contatore si ha sui fronti di discesa.

TABLE 4-1
Flip-Flop Characteristic Tables

(a) *JK* Flip-Flop

J	K	$Q(t+1)$	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

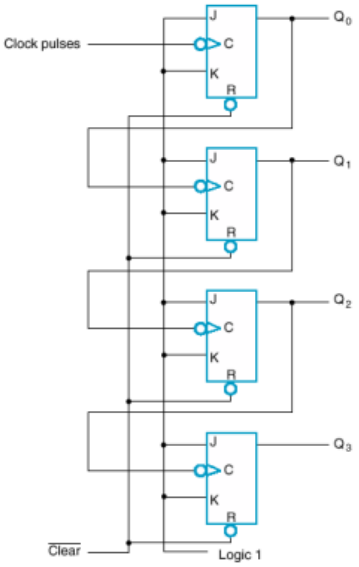
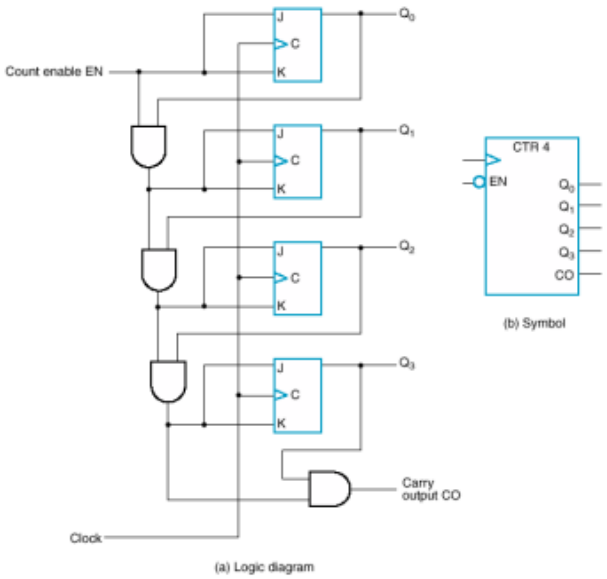


Fig. 5-8 4-Bit Ripple Counter

Mentre in un synchronous counter (contatore sincrono) l'ingresso C di tutti i FF è controllato da un clock comune. sono differenti dai ripple, in quanto l'ingresso di Clock è applicato a tutti i FF. Per JKFF la decisione di complementare l'uscita del FF è determinata dal valore degli ingressi J e K. Se J=K=0 il FF non cambia stato anche se è presente l'impulso di clock. Se J=K=1 lo stato prossimo è il complemento dello stato presente.



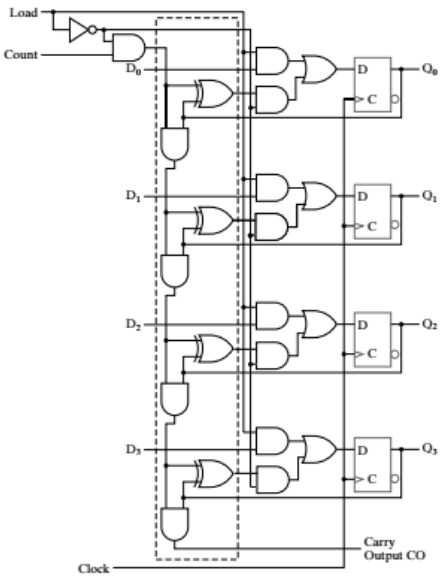
(a) Logic diagram

(b) Symbol

SINTESI DI UN CONTATORE: La procedura di sintesi di un contatore binario sincrono non è differente da quella di una qualsiasi rete sequenziale. Se non sono presenti ingressi esterni, la tabella di transizione di un contatore sincrono consta solo delle colonne del PS, del NS e dei Flip-flop.

Contatore binario con caricamento parallelo

- Lo schema mostra un registro a 4 bit con caricamento parallelo che ha anche la capacità di operare come contatore binario.
- Quando **Load=1** e **Count=0** viene disabilitato il conteggio e si abilita il caricamento parallelo dei dati.
- Se **Load=0** e **Count=1** il circuito opera da contatore binario.
- Se **Load=0** e **Count=0** il circuito mantiene il suo stato.
- **Carry Output** *(CO)* **=1** se tutti i FF sono nello stato di set e Count =1.



□ **FIGURE 14**
4-Bit Binary Counter with Parallel Load

PROM ed EPROM sono due tipi di memorie non volatili utilizzate per memorizzare dati permanenti o semi-permanenti in dispositivi elettronici. Ecco una descrizione dettagliata di ciascuno:

PROM= programmable read only memory, cioè ROM programmabili da noi una sola volta

EPROM= Erasable programmable read only memory, cioè ROM che possono essere cancellate, poiché hanno un “vetro”, la loro faccia, che sotto ultravioletto di dati si azzerano.

PAL, PLA, e FPGA sono tre tipi di dispositivi logici programmabili utilizzati per implementare funzioni logiche e circuiti digitali. Ecco una breve descrizione di ciascuno:

PAL= programmable Array Logic. è un tipo di dispositivo logico programmabile che utilizza un array programmabile di AND e un array fisso di OR per implementare funzioni logiche. Ha una flessibilità limitata rispetto agli altri due seguenti perché solo l’array di AND è programmabile.

PLA= programmable Logic Array. è un altro tipo di dispositivo logico programmabile che offre maggiore flessibilità rispetto al PAL, per cui può implementare funzioni logiche più complesse. Ha sia un array programmabile di porte AND che un array programmabile di porte OR.

FPGA= field Programmable Gate Array. è un dispositivo logico programmabile molto più complesso e flessibile rispetto a PAL e PLA. È composto da una matrice di blocchi logici programmabili, blocchi di memoria e interconnessioni programmabili. Ha una flessibilità estremamente elevata, permettendo la realizzazione di funzioni logiche molto complesse e circuiti digitali avanzati, inclusi processori interi

PROGRAMMAZIONE:

PUNTATORE: n puntatore è una variabile che contiene l'indirizzo di memoria di un'altra variabile. In altre parole, invece di contenere direttamente un valore, un puntatore "punta" alla posizione in memoria dove il valore è memorizzato.

Puntatori e array: I puntatori possono essere utilizzati per accedere agli elementi di un array.

VETTORI: un vettore è un array, ovvero una collezione di elementi dello stesso tipo memorizzati in posizioni contigue di memoria. Gli array permettono di gestire insiemi di dati usando un singolo nome di variabile e un indice per accedere agli elementi.

In certi contesti, un array può decadere in un puntatore al suo primo elemento. Questo accade quando si passa un array a una funzione o quando si assegna un array a un puntatore senza specificare la dimensione dell'array.

ALLOCAZIONE DINAMICA: L'allocazione dinamica della memoria permette di allocare memoria durante l'esecuzione del programma, piuttosto che durante la compilazione. Questo è utile quando la dimensione dei dati non è nota in anticipo o può cambiare nel tempo.

malloc: Alloca un blocco di memoria di una dimensione specificata in byte e restituisce un puntatore all'inizio del blocco. Non inizializza la memoria allocata, al contrario di calloc che inizializza tutti i bit della memoria allocata a 0.

calloc: Alloca memoria per un numero specificato di elementi di una dimensione specificata, inizializzando la memoria a zero.

realloc: Modifica la dimensione di un blocco di memoria precedentemente allocato.

free: Libera un blocco di memoria precedentemente allocato.

STACK: Lo stack è un'area di memoria utilizzata per l'allocazione automatica di variabili locali e per la gestione delle chiamate di funzione.

Caratteristiche:

- **Dimensione Fissa**: Solitamente ha una dimensione limitata e predeterminata.
- **Gestione Automatica**: La memoria viene allocata e deallocata automaticamente quando si entra e si esce dalle funzioni.
- **Velocità**: Molto veloce grazie all'allocazione/deallocazione semplice e lineare.
- **Ambito di Visibilità**: Le variabili allocate nello stack sono locali alla funzione in cui sono dichiarate.

HEAP: L'heap è un'area di memoria utilizzata per l'allocazione dinamica di memoria durante l'esecuzione del programma. Permette di allocare e deallocare memoria in modo esplicito usando funzioni come malloc, calloc, realloc e free.

Caratteristiche:

- **Dimensione Variabile**: La dimensione può crescere e ridursi dinamicamente durante l'esecuzione del programma.
- **Gestione Manuale**: La memoria deve essere allocata e deallocata manualmente dal programmatore.
- **Flessibilità**: Consente l'allocazione di blocchi di memoria di dimensioni variabili.
- **Efficienza**: Più lenta rispetto allo stack a causa della complessità dell'allocazione e deallocazione della memoria.

COMPILATORE: Un compilatore è un programma che traduce il codice sorgente scritto in un linguaggio di programmazione di alto livello (come C, C++, Java) in un linguaggio di basso livello (come il codice macchina o il linguaggio assembly) che può essere eseguito direttamente dal processore di un computer. Per esempio gcc è un insieme di compilatori, nato per il linguaggio c, e poi esteso per supportare una varietà di altri linguaggi

ARGV E ARGC: quando si parla di inserire dati "interattivamente da tastiera con l'uso di `argv` e `argc`" in un programma C, ci si riferisce al passaggio di parametri da linea di comando. Questo approccio è diverso dal leggere l'input direttamente durante l'esecuzione del programma con `scanf` o `fgets`.

• **`argc`**: È un intero che rappresenta il numero di argomenti passati al programma dalla linea di comando, incluso il nome del programma stesso.

• **`argv`**: È un array di stringhe (`char*`) che contiene gli argomenti passati al programma, dove `argv[0]` è il nome del programma stesso, e gli elementi successivi sono gli argomenti forniti dall'utente.

LIBRERIA: è una raccolta di funzioni predefinite e definizioni di tipi di dati che possono essere utilizzate da altri programmi.

CICLI: i cicli sono strutture di controllo che consentono di eseguire ripetutamente un blocco di istruzioni fino a quando una condizione specificata non è più verificata. I cicli sono fondamentali per automatizzare compiti ripetitivi e iterare su collezioni di dati. Ci sono principalmente tre tipi di cicli utilizzati in C: `for`, `while` e `do-while`.