Architettura degli Elaboratori
6 - Complementi sulle Reti Combinatorie

Ugo Dal Lago

Dipartimento di Scienze dell'Informazione
Università degli Studi di Bologna

Anno Accademico 2007/2008

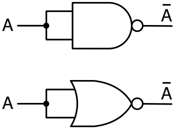
Sommario

Porte NAND e NOR

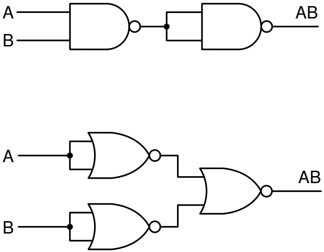
- Finora non abbiamo utilizzato molto le porte logiche **NAND** e **NOR**.
- Nelle implementazioni a due livelli c'è bisogno solo delle porte AND, OR, NOT.
  - Della porta NOT non c'è in realtà bisogno, perché si suppone di avere a disposizione due ingressi per ogni variabile: uno in cui la variabile è affermata e uno in cui è negata.
- Le porte NAND e NOR, a differenza delle altre, hanno un proprietà interessante: sono **universali**.
  - Ciò significa, in particolare, che ogni rete combinatoria può essere trasformata in una rete equivalente contenente solo la porta NAND o, in alternativa, contenente solo la porta NOR.
- Applicando le proprietà di De Morgan si possono ottenere simboli alternativi per le porte NAND e NOR:

NAND e NOR

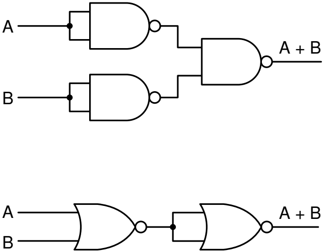
Tramite NAND e NOR si possono costruire reti combinatorie equivalenti alle porte NOT (in (a)), AND (in (b)) e OR (in (c)):



(a)



(b)



(c)

Universalità

- Esistono poi un certo numero di tecniche alternative per dimostrare l'**universalità** delle porte NAND e NOR, ossia per dimostrare che per ogni funzione booleana F esistono:
  - Una rete combinatoria contenente soltanto porte NAND.
  - Una rete combinatoria contenente soltanto porte NOR.
- Per quanto riguarda la porta NAND, si può procedere come segue:
  1. Prima di tutto si costruisce un'espressione E in somma di prodotti per F.
  2. Poi si costruisce la rete combinatoria a due livelli corrispondente a E.
  3. Si trasformano poi tutte le porte AND in porte NAND e tutte le porte OR in porte NAND.
- Per la porta NOR si procede in modo duale, costruendo un'espressione in prodotto di somme, la relativa rete a due livelli e trasformando porte OR e AND in porte NOR.
- Della **correttezza** di questa procedura ci si può rendere conto osservando il comportamento della rete combinatoria modificata.

Universalità

- Una **tecnica alternativa** è la seguente (per esempio nel caso della porta NAND):
  1. Si costruisce una rete combinatoria per la funzione F.
  2. Si rimpiazza ogni porta logica diversa da NAND presente nella rete con una rete combinatoria equivalente ad essa.

  Questa tecnica, però, produce una rete combinatoria avente un numero di porte logiche in generale maggiore rispetto a quello necessario nel caso della tecnica precedente.
- Esiste poi una **terza possibilità** (di seguito nel caso della porta NAND):
  1. Si costruisce una rete combinatoria (contenente solo porte AND e OR) per la funzione F.
  2. Si rimpiazza ogni porta AND con un NAND e ogni porta OR con un NAND (nella notazione alternativa).
  3. Si controlla che in ogni filo ci siano un numero pari di pallini e, in caso contrario, si inserisce una rete equivalente alla porta NOT lungo il filo.
- Entrambe le tecniche illustrate sono facilmente adattabili a tecniche che dimostrino l'universalità della porta NOR.

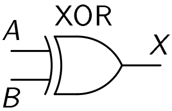
La Porta XOR

- Con **XOR** indicheremo sia una porta logica che un'operazione binaria sull'Algebra di Boole (quest'ultima indicata anche con $\oplus$).
- L'operazione $\oplus$ gode delle proprietà associativa e commutativa.
- Si può esprimere $\oplus$ nei termini delle operazioni di addizione, moltiplicazione e negazione come segue:

$$A \oplus B = A\overline{B} + \overline{A}B$$

- Intuitivamente, si può pensare che il risultato dell'applicazione di $\oplus$ a due cifre binarie valga 1 se e solo se esattamente uno dei due operandi vale 1.
  - Esistono quindi due configurazioni (tra le quattro possibili) in cui il risultato dell'applicazione di $\oplus$ vale 1.
- Il simbolo $\oplus$ diventerà anche parte del linguaggio per le espressioni booleane.

La Porta XOR



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$X = A \oplus B$$

Funzioni Pari e Dispari

- L'espressione booleana

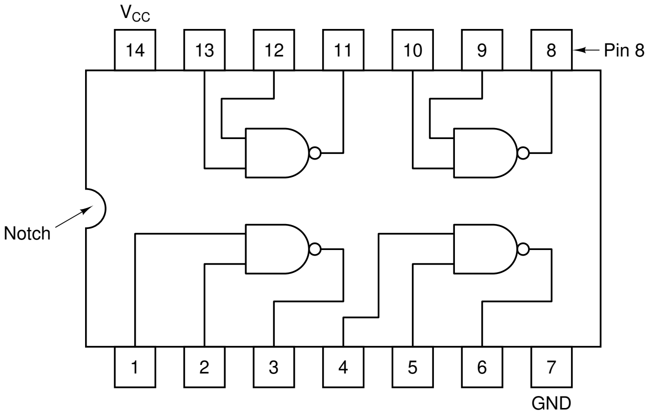
$$E_n = A_1 \oplus A_2 \oplus \ldots \oplus A_n.$$

  vale 1 solo quando il numero di variabili cui si assegna il valore 1 è dispari.
- La funzione di verità catturata dall'espressione E_n è detta **funzione dispari**.
- Se si osserva la mappa di Karnaugh per la funzione dispari, si nota subito che la sua implementazione in somma di prodotti (o in prodotto di somme) risulta estremamente inefficiente.
  - Serve una quantità esponenziale di porte AND, OR e NOT.
  - Utilizzando la porta XOR, basta invece una quantità lineare di porte.
- Il complemento della funzione dispari è detto **funzione pari**.
- Le funzioni per pari e dispari risultano molto utili per la generazione e il controllo di parità.

Circuiti Integrati

- Le porte logiche non sono vendute individualmente, ma in unità chiamate **circuiti integrati** (cui spesso ci si riferisce con i termini **IC** oppure **chip**).
- Un IC è un quadrato di silicio di lato pari a $5mm$, su cui sono posizionate alcune porte logiche.
- Uno o più IC sono poi inseriti in contenitori rettangolari di plastica o ceramica.
- I chip possono essere approssimativamente classificati in base al numero di porte che contengono:
  - Circuiti **SSI**: da 1 a 10 porte.
  - Circuiti **MSI**: da 10 a 100 porte.
  - Circuiti **LSI**: da 100 a 100.000 porte.
  - Circuiti **VLSI**: più di 100.000 porte.
- Per tutti i nostri scopi le porte logiche sono **ideali**, nel senso che l'output appare non appena viene applicato il segnale in input.
  - In realtà i chip hanno un ritardo temporale finito, chiamato **ritardo della porta**.

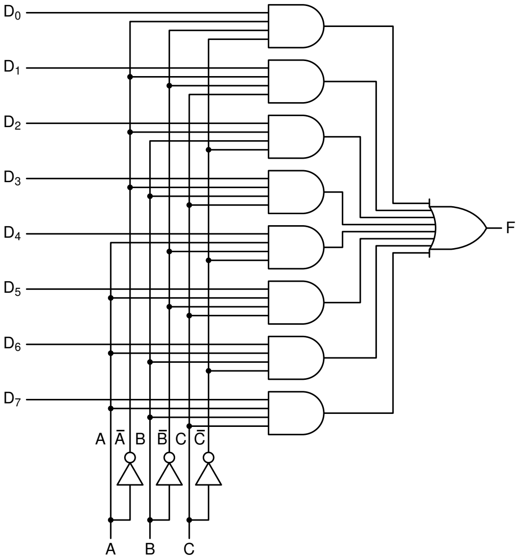
Chip SSI: Esempio



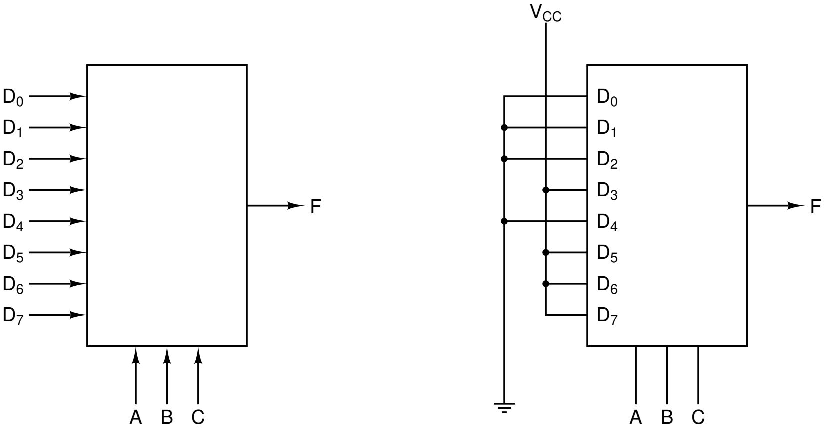
Multiplexer

- Un **multiplexer** è una rete combinatoria con:
  - 2^n ingressi per i dati.
  - n ingressi di controllo.
  - Una singola uscita.
- Gli input di controllo permettono di **selezionare** uno tra i possibili ingressi per i dati, il cui input diventerà il valore dell'unica uscita.
  - Ad ogni sequenza di valori di verità lunga n corrisponderà un numero binario compreso tra 0 e $2^n - 1$.
- Assegnando valori costanti agli ingressi per i dati, si può costruire una rete combinatoria che implementi qualunque funzione booleana di n variabili.
- Un'altra applicazione del multiplexer è la conversione di dati da parallelo a seriale.
- Il **demultiplexer** redirige il suo unico ingresso per i dati verso una delle sue 2^n uscite in base ai valori degli ingressi di controllo.

Multiplexer: Esempio



Multiplexer: Esempio

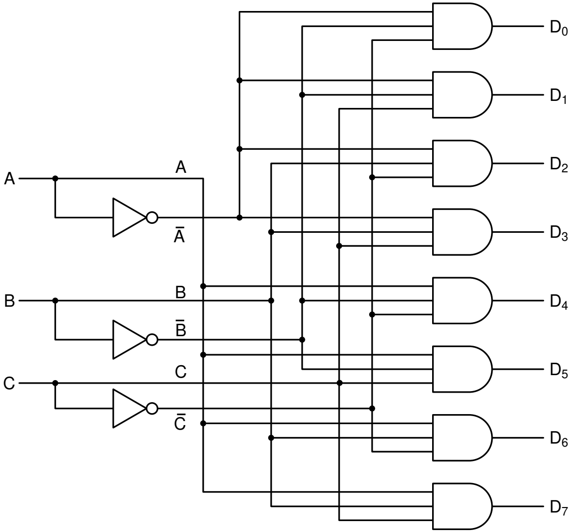


(a) (b)

Decodificatori

- Un **decodificatore** è una rete combinatoria con:
  - n ingressi.
  - 2^n uscite.
- I valori in ingresso vengono utilizzati per selezionare una tra le uscite, che sarà l'unica ad essere impostata ad 1.
- Una possibilie applicazione del codificatore è la seguente:
  - Supponiamo di avere a disposizione una memoria principale da 8 MB, suddivisa in 8 chip da 1 MB ciascuno.
  - Per indirizzare ognuno dei byte nella memoria, occorre avere a disposizione indirizzi da 23 bit, mentre ognuno degli 8 chip lavora con indirizzi a 20 bit.
  - I tre bit più significativi saranno utilizzati per selezionare il chip cui si riferisce l'indirizzo di partenza tra gli 8 possibili.
  - In quest'ultima fase un decodificatore risulterà molto utile.
- Dualmente al decodificatore avremo il **codificatore**, con 2^n ingressi e n uscite.
  - L'inerente ambiguità viene in genere risolta tramite un meccanismo a **priorità**.

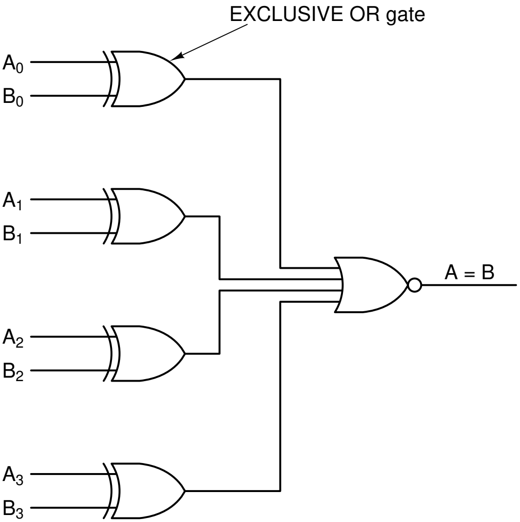
Decodificatori: Esempio



Comparatori

- Un **comparatore** è una rete combinatoria con:
  - $2n$ ingressi, suddivisi in due sequenze lunghe n.
  - Un'uscita.
- Il valore dell'unica uscita sarà 1 se le due sequenze in input sono identiche, mentre sarà 0 se le due sequenze sono diverse.
  - In alternativa, si può fare in modo che l'uscita valga 0 se le due sequenze sono diverse e 1 se le due sequenze sono diverse.
- La porta logica XOR risulterà molto utile nella realizzazione del comparatore.
- Oltre all'uscita che indica se le due sequenze sono uguali, possono essere presenti **altre uscite**, che indicano se le due sequenze sono una maggiore dell'altra (qualora interpretate come numeri binari).

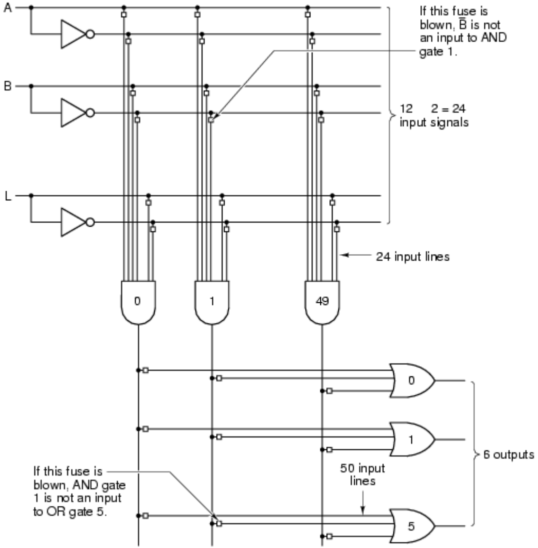
Comparatori: Esempio



Array Logici Programmabili (PLA)

- L'**Array Logico Programmabile** (PLA) è un chip molto generale, che permette di calcolare somme di prodotti.
- Diversamente dalle reti descritte finora, il PLA include un certo numero di **fusibili**, che possono essere bruciati per determinare la funzione da calcolare.
- Al momento della fabbricazione, i fusibili sono intatti.
- L'array logico programmabile include:
  - k ingressi.
  - k porte logiche NOT.
  - n porte logiche AND.
  - m porte logiche OR.
- In questo modo, m somme di al più n prodotti di al più k letterali potranno essere implementate tramite un singolo PLA.
- Anche se i PLA programmabili sul campo sono utilizzati ancora oggi, si preferisce impiegare dei PLA **costruiti ad hoc**.

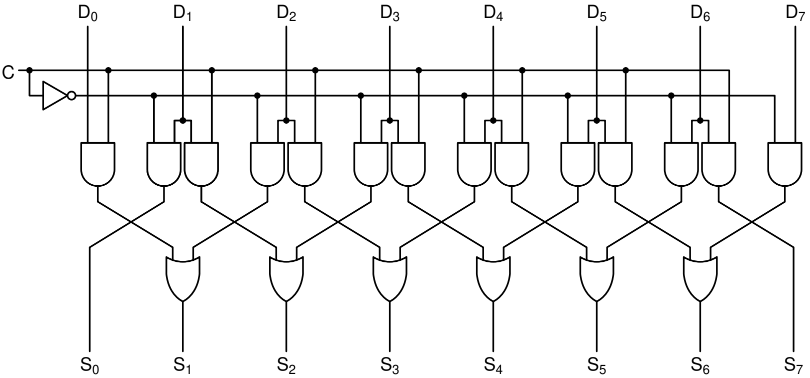
Array Logici Programmabili (PLA): Esempio



Registri a Scorrimento

- Il **Registro a Scorrimento** è una rete combinatoria con
  - n ingressi per i dati.
  - Un ingresso di controllo.
  - n uscite.
- L'idea è quella di far scorrere **verso destra** o **verso sinistra** i valori in input.
  - L'input di controllo serve proprio a determinare se lo scorrimento debba avvenire verso destra o verso sinistra.
- In ognuno dei due casi, il valore di uno tra gli ingressi andrà perso, mentre una tra le uscite assumerà un valore non proveniente dagli ingressi e fissato a priori.

Registri a Scorrimento: Esempio

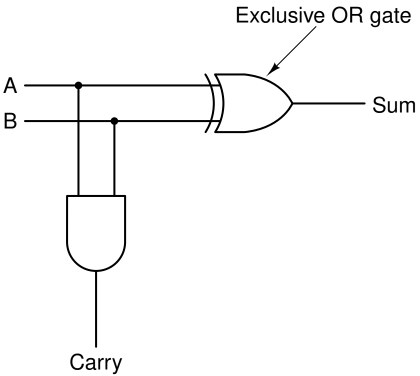


Semisommatore (Half-Adder)

- Un **semisommatore** (o **half-adder**) è una rete combinatoria con:
  - Due ingressi.
  - Due uscite.
- La prima delle due uscite è valorizzata con la **somma** dei due bit in ingresso, mentre la seconda è valorizzata con il **riporto** generato dalla somma dei due bit in ingresso.
- Non è possibile utilizzare il semisommatore per calcolare la somma (e il riporto relativo al bit più significativo) di una sequenza di n bit (con $n \geq 1$).
  - In particolare, manca la possibilità di **propagare** il riporto ai bit più significativi.

Semisommatore (Half-Adder)

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



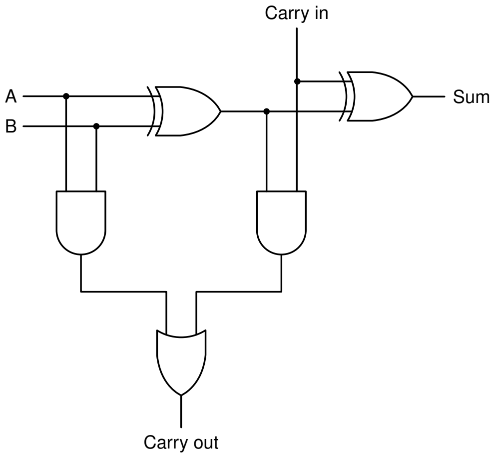
Sommatore (Full-Adder)

- Un **sommatore** (o **full-adder**) è una rete combinatoria con:
  - Tre ingressi.
  - Due uscite.
- Come nel caso del semisommatore, la prima delle due uscite è valorizzata con la **somma** dei bit in ingresso (che in questo caso sono tre, due operandi e un **riporto in entrata**), mentre la seconda è valorizzata con il **riporto** generato dalla somma dei bit in ingresso.
- È facile rendersi conto che un sommatore può essere realizzato a partire da due semisommatori.

Sommatore (Full-Adder)

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)

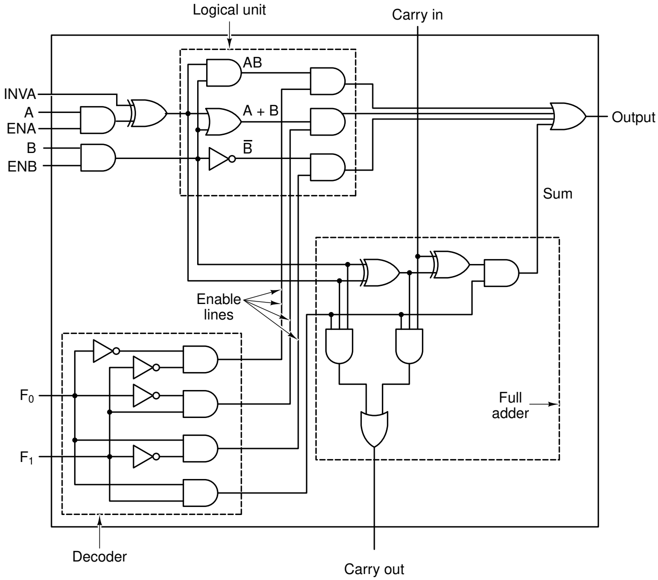
Propagazione di Riporto e Selezione di Riporto

- I sommatori, al contrario dei semisommatori, possono essere combinati a formare un sommatore a n bit.
  - Per ogni bit, il riporto in uscita viene utilizzato come riporto in entrata per il bit successivo.
- Parliamo in questo caso di **sommatore a propagazione di riporto**.
  - Se supponiamo che ogni porta logica aggiorni il suo output con un ritardo costante, la somma di n bit richiederà tempo proporzionale a n.
- Il ritardo diventa logaritmico in n se si utilizza un **sommatore a selezione di riporto**.
  - L'idea è quella di calcolare preventivamente le somme dei bit più significativi sia nel caso in cui il relativo riporto sia 1, sia nel caso in cui sia 0.

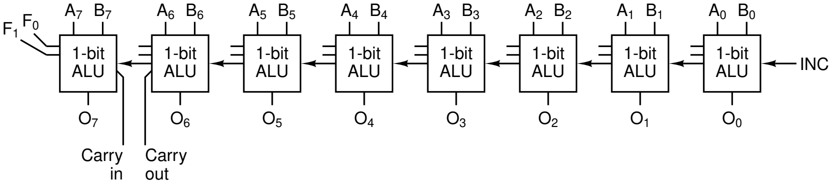
Unità Aritmetico-Logiche

- Come sappiamo, la CPU contiene un'unità funzionale, chiamata **ALU** o **Unità Aritmetico-Logica**, in grado di effettuare sia operazioni logiche che aritmetiche.
- L'unità artimetico-logica è una rete combinatoria con:
  - n ingressi per i dati.
  - m ingressi di controllo.
  - n uscite.
- Gli ingressi di controllo servono a selezionare quale tra le operazioni possibili vada applicata ai dati.
- Spesso le unità aritmetico-logiche consistono in n piccole reti combinatorie, ognuna delle quali agisce su un singolo bit di dati ma riceve in ingresso tutti gli m bit di controllo.
  - In tal caso, occorrerà prevedere che ognuna delle reti abbia un **ingresso** supplementare per il riporto in ingresso e un un'**uscita** supplementare per il riporto in uscita.

Una ALU a 1 Bit



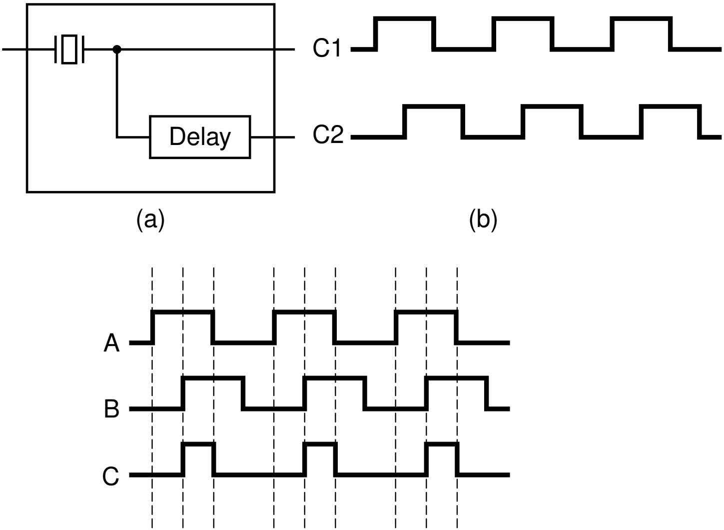
Una ALU a 8 Bit



Clock

- Molti circuiti digitali utilizzano un **clock** per sincronizzarsi e per ottenere le informazioni temporali desiderate.
- Un clock è un circuito che emette una serie di impulsi di larghezza definita.
- La frequenza degli impulsi è in genere compresa tra 1 e 500 Mhz, corrispondenti a cicli di clock compresi tra 1000 e 2 nanosecondi.
- Inserendo un segnale proveniente da un clock in un circuito che non modifica il segnale ma che applica un ritardo predefinito, si può ottenere un secondo segnale di clock, **traslato** rispetto al primo.
- Segnali provenienti da clock diversi possono essere combinati tra loro a formare **clock asimmetrici**.

Clock, Diagramma di Temporizzazione, Clock Asimmetrico



(a) (b)

(c)