giacomo alessandroni

introduzione ai circuiti combinatori

indice generale

logica combinatoria

introduzione

L'elettronica digitale è una branca dell'elettronica: si occupa di gestire ed elaborare informazioni in formato numerico[1] (o discreto).

È possibile distinguere due categorie di logica: **combinatoria** e **sequenziale**. Nella logica combinatoria le uscite dipendono esclusivamente dagli ingressi. Nella logica sequenziale le uscite dipendono dagli ingressi e dalla "storia" (meglio dallo *stato*) del sistema. Il sistema sequenziale ha quindi una memoria: unita all'ingresso restituisce l'uscita.

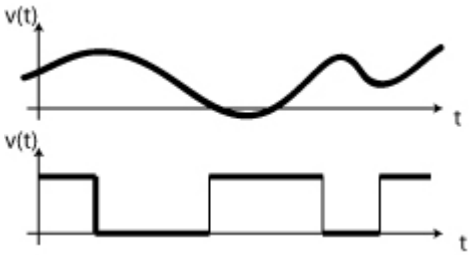
Due esempi per comprendere le differenze tra queste due categorie:

1. il circuito "pilota" di un display a sette segmenti deve "tradurre" il numero da visualizzare in segmenti da illuminare. Questa operazione non dipende dalla sequenza dei numeri, pertanto l'uscita dipende solo dall'ingresso;
2. l'ascensore – per il suo corretto funzionamento – necessita di una memoria. Chiamandolo ad un dato piano possono verificarsi tre eventi differenti:
   - 2.1 se si trova al piano richiesto deve soltanto aprire la porta;
   - 2.2 se è a un piano inferiore deve salire di tanti piani quanto necessario;
   - 2.3 se si trova ad un piano superiore dovrà scendere.

Tutto questo – naturalmente – implica una memoria per sapere in quale piano si trova l'ascensore.

dall'elettronica analogica all'elettronica numerica

L'elettronica numerica lavora con circuiti e sistemi operanti solo su due stati, come due livelli di tensione o di corrente. Nei sistemi digitali i due stati – combinati tra loro – rappresentano numeri, lettere, colori e qualsiasi informazione. Nei sistemi numerici a due stati, detti *sistemi binari*, le due cifre sono 0 ed 1[2]. Ognuna di esse costituisce un *bit*.



In figura sono riprodotti due segnali: il primo analogico (può assumere tutti i valori compresi in un intervallo), il secondo numerico (assume solo due valori prefissati). Entrambi sono soggetti ad un rumore di fondo e tutti ne abbiamo fatto esperienza:

1. le vecchie musicassette – oltre al segnale da riprodurre – avevano con sé il rumore di fondo. Il rumore risultava evidente quando la cassetta veniva duplicata più volte. Questo fatto introduce un limite nel trattamento dei segnali;
2. nei *compact disc* – per non cambiare argomento – la musica è memorizzata in formato numerico. Il rumore non è scomparso (magari!), ma ci sono diverse tecniche per affrontarlo. Le principali:
   - 2.1 definizione di un valore di soglia per la determinazione del *livello logico alto* e del *livello logico basso*. Fatto ciò si determina cosa deve essere riprodotto in uscita;

1 Molti testi usano la dicitura *elettronica digitale*, derivante dall'inglese *digit* (cifra), figlia del latino *digitus* (dito). Per questo motivo si preferisce utilizzare la locuzione *numerica*.

2 I livelli 1 e 0, nei testi, come nei manuali, possono essere sostituiti da *alto* e *basso*; *vero* e *falso*; *bianco* e *nero* e così via finché c'è fantasia a disposizione.

2.2 è possibile aggiungere un codice di controllo in coda all'informazione, si fa un test: se non vi è la corrispondenza attesa, si procede ad una seconda lettura.

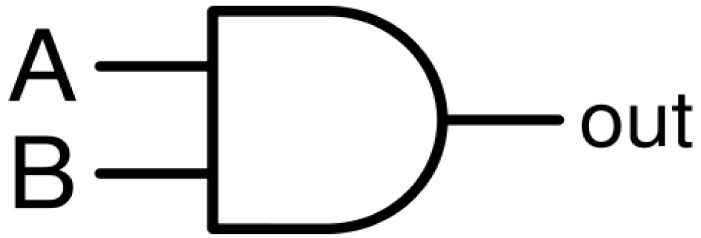
porte logiche

Le porte logiche sono circuiti elettronici: eseguono operazioni con due soli livelli di tensione. Di seguito vengono elencate le porte logiche più comuni.

porta and

La porta logica AND è composta di due ingressi e un'uscita. Restituisce valore logico 1 se entrambi gli ingressi hanno valore 1.

Da questa definizione si ottiene la sua *tabella della verità*: la correlazione ingressi/uscita.

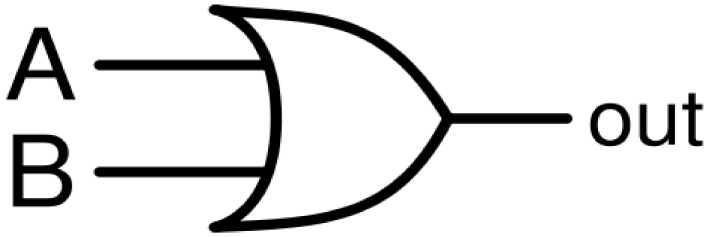
Simbolo della porta logica AND	Tabella della verità		
	A	B	out
	0	0	0
	0	1	0
	1	0	0
	1	1	1

Nel calcolo viene rappresentata come una moltiplicazione. Si scrive quindi:

$$out = A \cdot B$$

porta or

La porta logica OR è composta di due ingressi e un'uscita. Restituisce valore logico 1 se almeno un ingresso ha valore 1.

Simbolo della porta logica OR	Tabella della verità		
	A	B	out
	0	0	0
	0	1	1
	1	0	1
	1	1	1

Nel calcolo viene rappresentata come una somma. Si scrive quindi:

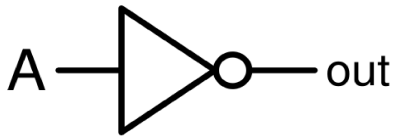
$$out = A + B$$

porta not

La porta logica NOT[3] è composta da un ingresso e un'uscita. Restituisce il valore opposto a quello in ingresso.

Nei circuiti logici, per non appesantire gli schemi, la negazione è spesso rappresenta con un singolo cerchietto.

3 In alcuni testi, in particolare quelli anglosassoni, viene chiamata anche *inverter*.

Simbolo della porta logica NOT	Tabella della verità	
	A	out
	0	1
	1	0

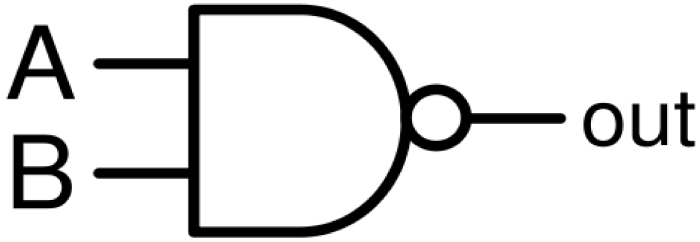
Nel calcolo viene rappresentata con un tratto sopra l'ingresso. Si scrive quindi:

$$out = \overline{A}$$

porta nand

La porta logica NAND è composta di due ingressi e un'uscita. Restituisce valore logico 1 se almeno un ingresso ha valore 0.

La porta NAND è una *porta logica universale*: qualsiasi altra porta logica può essere ottenuta dalla opportuna combinazione di porte NAND.

Simbolo della porta logica NAND	Tabella della verità		
	A	B	out
	0	0	1
	0	1	1
	1	0	1
	1	1	0

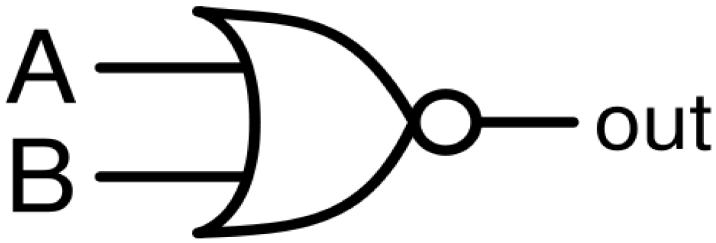
Nel calcolo viene rappresentata come una moltiplicazione, seguita da una negazione. Si scrive quindi:

$$out = \overline{A \cdot B}$$

porta nor

La porta logica NOR è composta di due ingressi e un'uscita. Restituisce valore logico 1 se entrambi gli ingressi hanno valore 0.

Anche la porta NOR è una porta logica universale.

Simbolo della porta logica NOR	Tabella della verità		
	A	B	out
	0	0	1
	0	1	0
	1	0	0
	1	1	0

Nel calcolo viene rappresentata come una somma, seguita da una negazione. Si scrive quindi:

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

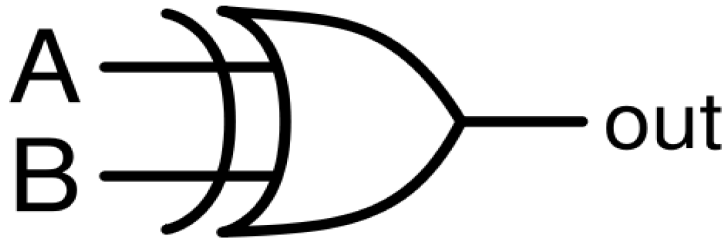
porte xor e xnor

Due altre porte logiche sono l'OR-esclusivo (o X-OR) e la sua funzione inversa, il NOR-

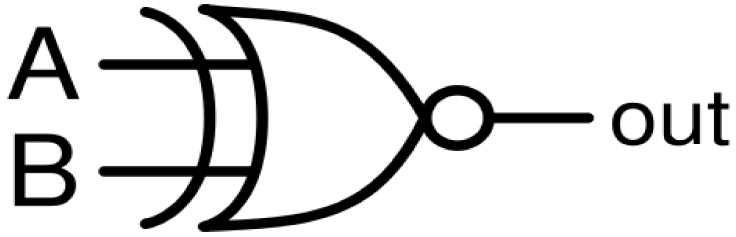
esclusivo (o X-NOR).

L'OR-esclusivo ha uscita 1 se i due valori di ingresso sono diversi. Se ci sono più di due ingressi, l'uscita è 1 se il numero di segnali 1 al suo ingresso è dispari.

Viceversa il NOR-esclusivo ha uscita 1 se i due valori di ingresso sono uguali. Se ci sono più di due ingressi, l'uscita è 1 se il numero di segnali 1 al suo ingresso è pari.

Simbolo della porta logica XOR	Tabella della verità		
	A	B	out
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Nel calcolo si utilizza questo simbolo: $out = A \oplus B$

Simbolo della porta logica XNOR	Tabella della verità		
	A	B	out
	0	0	1
	0	1	0
	1	0	0
	1	1	1

Analogamente, per il NOR-esclusivo si ricorre a: $out = \overline{A \oplus B}$

il nand può tutto

Veniamo ora alle porte logiche universali e dimostriamo questa cosa per la più usata.

Porta logica		Struttura NAND equivalente
AND		
OR		
NOT		
NOR		

Inutile continuare. La cosa importante da sapere è che se una porta logica può sintetizzare tutte le altre, può essere usata come *unica* porta logica.

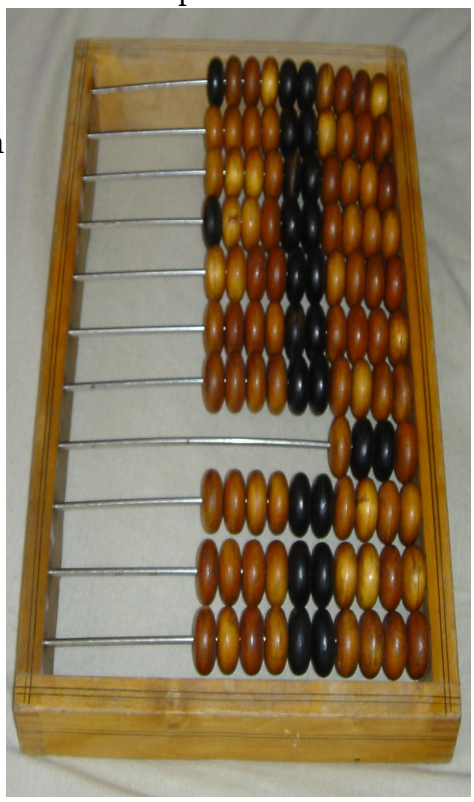
respiriamo un poco

Uno, due, tre, quattro, cinque. Una mano. Sei, sette, otto, nove, dieci. Due mani. Una mano viene stilizzata con una “V”. Due mani con due V contrapposte, cioè una “X”. Per ogni singolo dito, ovviamente, basta scrivere una “I”.

Questo è il *nostro* modo di contare. Nostro non solo perché questi numeri fanno ancora mostra di sé sui monumenti e nei testi letterari (Dante Alighieri contava ancora così). Ma perché nasce proprio da un nostro *imprinting* molto antico. La nostra, in origine, nasce come una civiltà molto individuale, in cui la casa conta molto più della piazza o del *bazar*. Lo scambio con gli altri individui è basato essenzialmente sullo *status*, non sul commercio.

Altre civiltà (in genere quelle orientali) nascono invece fin dall’inizio coi grandi numeri – le decine, le centinaia, le migliaia – e hanno bisogno di lavorare rapidamente con essi, non di analizzare a fondo ogni singola (antropomorfica) unità. Un individuo di queste società, di per sé, conta poco ed è inserito da subito in una situazione di massa, un impero idraulico o qualcosa del genere. Non è quasi mai autosufficiente dal punto di vista alimentare ed ha subito bisogno di una rete molto complessa di scambi: il commercio. Per gestire

quest'ultimo gli farebbe molto comodo un computer e – non trovandolo al bazar – è costretto a inventarsi qualcosa il più vicino possibile, un abaco ad esempio.



L'abaco funziona già per strutture logiche, per posizionamenti qualitativi e non per semplice addizione di unità: la colonna delle decine è completamente diversa da quella delle unità ed ha bisogno di un software logico, non di una mera percezione fisica, per funzionare (contare sulle dita è usare semplicemente un hardware). Il passaggio successivo – naturalmente – è l'invenzione dello zero.

Tutto qua? No. I fenici e i greci, in un punto di snodo fra est e ovest, inventano – ma su un *target* del tutto differente – un altro software molto interessante: l'alfabeto. Il successo di vendita di quest'ultimo è enorme: viene immediatamente sviluppata una *release* per consentire di utilizzarlo anche per i calcoli numerici (*alpha* vale uno, *beta* vale due, ecc.), una specie di Office (Word + Excel) di quei tempi.

Questo programma ha così successo e per un bel pezzo (grazie anche a un'accorta strategia di compatibilità col vecchio *Conta-sulle-dita 3.1*) copre tutto il mercato. Solo quando l'azienda produttrice, la Impero & C, va a ramengo (ma ci vorranno un bel po' di secoli) la concorrenza orientale potrà finalmente piazzare la tecnologia *Zero-based* anche dalle nostre parti. Per questo ci è toccato correre molto alla svelta per recuperare il *gap*. Sempre ammesso che l'abbiamo *veramente* recuperato.

bit, byte, megabyte... (passando per il nibble)

Fatta la pausa, è importante chiarire a cosa servono questi strumenti. Si utilizzano per la manipolazione delle unità di informazione, i famosi *bit*. Un bit è un'informazione: può valere solo 0 oppure 1. Da solo serve a pochissimo.

Accade lo stesso con i nostri numeri facendo corrispondere al bit l'*unità*. Con i simboli 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (e null'altro) è possibile rappresentare un insieme di dieci oggetti (sfruttando anche lo 0), ma nulla di più. Il nostro infatti è un sistema *decimale ordinale*: 10, 100 e 1000 sono tre numeri diversi solo per la posizione (l'ordine) del numero 1.

È possibile fare lo stesso con soli due numeri: il sistema binario[4] è un sistema numerico ordinale in base 2. Utilizza due simboli: 0 e 1, invece dei dieci del sistema decimale tradizionale. Di conseguenza, la cifra in posizione N (da destra) si considera moltiplicata per 2^N (anziché per 10^N come avviene nella numerazione decimale).

Questa tabella confronta le rappresentazioni binarie, esadecimali (con sedici simboli) e decimali di alcuni numeri, limitatamente a 4 bit, una struttura ancora molto utilizzata nell'elettronica di base.

4 È considerato tra le più grandi invenzioni del matematico tedesco Gottfried Wilhelm von Leibniz; essa cadde nel vuoto e solo nel 1847 verrà riscoperta, grazie al matematico inglese George Boole: aprirà l'orizzonte alle grandi scuole di logica matematica del '900 e – soprattutto – alla nascita del calcolatore elettronico.

binario	esadecimale	decimale
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Il sistema binario utilizza tutti i suoi simboli, proprio come il sistema decimale. Quando li termina (essendo solo due la cosa capita spesso) si aggiunge un elemento nella seconda colonna e così via.

Nel 1971 Federico Faggin (italiano!) sviluppò e diresse il progetto del primo microprocessore, l'Intel 4004, contribuendo con idee fondamentali alla sua realizzazione. Questo microprocessore lavorava a gruppi di 4 bit, denominati *nibble*, ormai preistoria.

Un *byte*[5] è una sequenza di 8 bit, in grado di assumere $2^8 = 256$ valori.

un chilo non è 1000

I multipli del byte sono sì il *kilobyte* e via dicendo, ma un chilo non è mille come al mercato dove regna sovrano il sistema decimale. Nel sistema binario 1kb = 2^{10}b = 1024b. Se considerassimo 1000b commetteremmo un errore del 2,4%. I multipli infatti non si ottengono con 10^3, 10^6, 10^9, ecc. bensì con 2^{10}, 2^{20}, 2^{30}, ecc.

Vediamo cosa succede:

Fattore	Quantità	Nome	Sigla	Errore
2^0	1	byte	b	0,00%
2^{10}	1.024	Kilobyte	Kb	2,24%
2^{20}	1.048.576	Megabyte	Mb	4,86%
2^{30}	1.073.741.824	Gigabyte	Gb	7,37%
2^{40}	1.099.511.627.776	Terabyte	Tb	9,95%
2^{50}	1.125.899.906.842.624	Petabyte	Pb	12,59%
2^{60}	1.152.921.504.606.846.976	Exabyte	Eb	15,29%

La colonna **Errore** non è stata inserita per far studiare qualcosa in più. Vi interessa! È la percentuale di memoria (soprattutto hard-disk e memorie usb) venduta in meno con furbizia. I dischi rigidi ormai hanno capienze dell'ordine di grandezza del Terabyte, ma il costruttore – quando dichiara un Terabyte – fa riferimento al *Sistema Internazionale* e rifila 10^{12}byte < 2^{40}byte (controllate a casa vostra, se non credete).

5 Contrazione di **bi**nary **t**erm, elemento binario

le mappe di karnaugh

Le mappe di Karnaugh sono uno strumento potente per passare da una funzione logica ad un circuito vero è proprio.

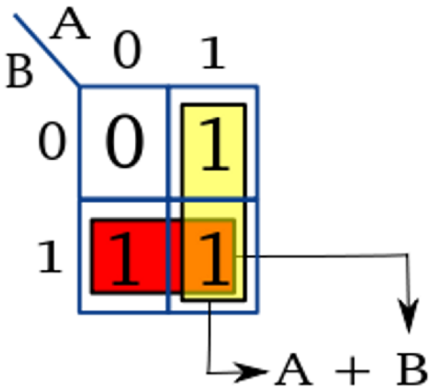
Prima cosa da ricordare: sono una rappresentazione di una funzione booleana per evidenziare le coppie di *mintermini* (raggruppamenti di 1) o di *maxtermini* (raggruppamenti di 0) a distanza di *Hamming* unitaria (cambiando un solo bit alla volta). Per costituzione risultano applicabili fino a funzioni di 5/6 bit.

Si tratta di un metodo grafico: si riporta la tabella della verità su un quadrato (o rettangolo se il numero dei bit in ingresso è dispari), si dividono i bit in due parti (colonne e righe) e si scrivono così che vari un solo bit alla volta (00, 01, 11, 10 è un esempio classico, ma qualsiasi altro ordine – se rispetta questa regola – è benvenuto).

Quello che si ottiene è sì un quadrato, ma i suoi estremi si congiungono sempre. La sua migliore definizione è sfera. Vediamo ora una tabella della verità, come si costruisce ed usa la sua mappa di Karnaugh.

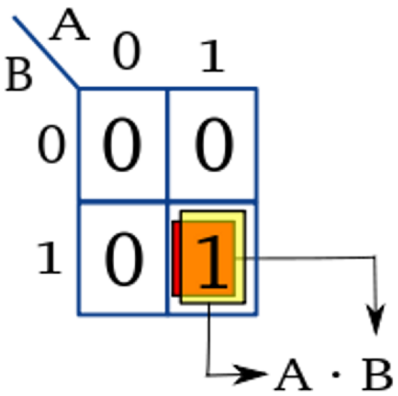
la mappa di karnaugh a 2 bit

Le mappe di Karnaugh a 2 variabili sono banali, ma possono essere usate come introduzione per comprenderne il funzionamento. Le mappa delle porte OR ed AND a 2 ingressi sono queste.



All'interno della mappa di Karnaugh sono riportate le uscite. All'esterno le entrate, in questo caso A e B.

La mappa di Karnaugh della porta OR è completata da soli valori logici '1' tranne nella cella in alto a sinistra. All'interno della prima mappa, le celle adiacenti contenenti '1' vengono raggruppate a coppie, a gruppi di quattro, o otto, in ogni caso per multipli di 2^N.



In questo caso, vi è uno gruppo orizzontale e verticale di due. Si indicano i raggruppamenti disegnando un ellisse o un rettangolo per ciascuno.

Il gruppo orizzontale B corrisponde a un valore di 1. Nella cella di sinistra A = 0 e nella destra A = 1. In altre parole, il valore di A non pregiudica il risultato dell'espressione booleana per queste celle. Senza raggruppare, si sarebbe scritta l'espressione booleana così:

$$\bar{A} \cdot B + A \cdot B$$

Dopo il raggruppamento questo si riduce a *B*, cosa che – tra qualche capitolo – dimostreremo anche analiticamente, ma difficile da notare a colpo d'occhio.

Analogamente, il raggruppamento verticale – se si considerasse una cella alla volta – produrrebbe quest'espressione:

$$A \cdot \bar{B} + A \cdot B$$

La quale, una volta semplifica, da luogo ad *A*. Non è difficile intuire che – per soddisfare tutte le condizioni che generano uscita 1 – è necessario inserire entrambi i raggruppamenti sommandoli, ottenendo $A + B$ che altri non è che l'espressione booleana di un OR.

dalla tabella della verità alla mappa di karnaugh

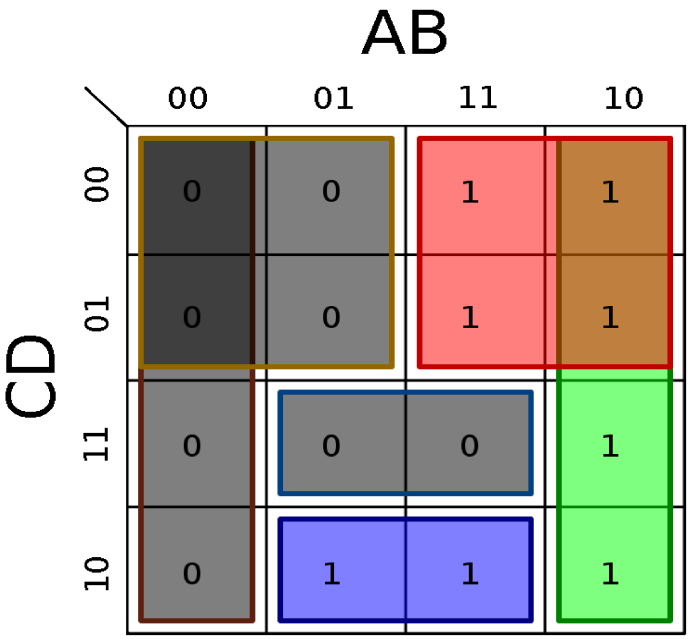
Consideriamo una genericafunzione con 4 bit in ingresso.

EX	A	B	C	D	out
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
A	1	0	1	0	1
B	1	0	1	1	1
C	1	1	0	0	1
D	1	1	0	1	1
E	1	1	1	0	1
F	1	1	1	1	0

Essendo 16 le combinazioni delle 4 variabili booleane, anche la mappa di Karnaugh dovrà avere 16 posizioni. Il modo più conveniente per disporle è in una tabella 4x4.

Per come è stata impostata la mappa, i numeri scorrono in verticale: le due cifre più significative sono nelle colonne, le altre due (quelle meno significative) nelle righe. Da notare che le coppie di variabili di ingresso (AB e CD) sono ordinate con un codice di Hamming: così fra coppie di celle adiacenti si modifica una sola variabile per volta.

Dopo aver costruito la mappa di Karnaugh si raggruppano gli 1 in rettangoli più grandi possibili, il vincolo è che si tratti di una potenza di 2 (2, 4, 8 celle) non altro. I raggruppamenti ottimali, in questo esempio, sono quelli indicati nella mappa.



Per ciascun raggruppamento si trovano le variabili a valore costante. Per il primo raggruppamento (il rosso) si ha:

- la variabile A mantiene lo stesso stato (1) in tutto il gruppo, quindi deve essere inclusa nel prodotto risultante;
- la variabile B non mantiene il suo valore, passando da 1 a 0, quindi deve essere esclusa;

- C non cambia (0) e quindi viene inclusa;
- D cambia ed è esclusa.

Così il primo prodotto dell'espressione booleana finale è $A \cdot \overline{C}$.

Per il rettangolo in verde si vede che A e B mantengono lo stesso stato, mentre C e D cambiano. Essendo B pari a 0, la variabile deve essere negata prima di venire inclusa nel prodotto. Così si ottiene $A \cdot \overline{B}$.

Con lo stesso procedimento si trova $B \cdot C \cdot \overline{D}$ dal rettangolo blu (più piccolo è il raggruppamento, maggiori sono i termini).

L'espressione finale si ottiene sommando i prodotti precedentemente trovati.

$$out = A \cdot \overline{C} + A \cdot \overline{B} + B \cdot C \cdot \overline{D}$$

Se si vuole trovare la funzione duale, ossia la funzione che fa uso dei *maxtermini*, la procedura è

- raggruppare gli 0 in luogo degli 1;
- sommare i termini anziché moltiplicarli (negando gli 1, anziché gli 0);
- infine eseguire il prodotto dei termini ottenuti.

Ciò corrisponde nello scegliere le righe dove l'uscita vale 0. In questo caso la tabella della verità è stata costruita ad arte, ma può capitare di ritrovarsi un solo 0 ed a quel punto è evidente: conviene raggruppare lui.

Raggruppando i mintermini si ottiene la funzione:

$$out = (A + C) \cdot (A + B) \cdot (\overline{B} + \overline{C} + \overline{D})$$

un'applicazione pratica

È possibile eseguire la conversione di un numero decimale in un altro codice: verrà usato per pilotare un display a 7 segmenti, questi sono – appunto – le uscite di un decodificatore da BCD[6] a 7 segmenti. La tabella della verità mostra quali segmenti devono essere illuminati (1 logico) per rappresentare la cifra decimale desiderata.

La codifica dei numeri decimali in binario necessita di 4 bit per ogni cifra decimale da 0 a 9[7]. Tale codifica rappresenterà gli ingressi del nostro decodificatore da BCD a 7 Segmenti.

tabella della verità

In questo caso si hanno quattro ingressi (**A**, **B**, **C** e **D**) e sette uscite (**a**, **b**, **c**, **d**, **e**, **f** e **g**). Per questo motivo si risolveranno sette mappe di Karnaugh. Tranquilli: non sarà impresa ardua e disperatissima.

6 La codifica BCD (*Binary-Coded Decimal*) è un protocollo comunemente utilizzato in informatica ed elettronica per rappresentare le cifre decimali in codice binario.

7 Infatti $2^3 = 8$ e non sarebbe sufficiente. $2^4 = 16$ e – anche se c'è dell'avanzo – pazienza.

A	B	C	D	DEC	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	1	0	1	1	0	0	0	0
0	0	1	0	2	1	1	0	1	1	0	1
0	0	1	1	3	1	1	1	1	0	0	1
0	1	0	0	4	0	1	1	0	0	1	1
0	1	0	1	5	1	0	1	1	0	1	1
0	1	1	0	6	1	0	1	1	1	1	1
0	1	1	1	7	1	1	1	0	0	0	0
1	0	0	0	8	1	1	1	1	1	1	1
1	0	0	1	9	1	1	1	1	0	1	1

Infatti le combinazioni esadecimali **A**, **B**, **C**, **D** ed **F** sono *indifferenti*: questi ingressi non sono previsti: possiamo assegnargli il valore che più fa comodo, per semplificare il circuito logico.

Le condizioni di indifferenza possono essere considerarle 0 o 1 a piacere. Questo perché gli ingressi superiori a 9 non sono previsti (le cifre decimali vanno da 0 a 9). Questo semplificherà la ricerca delle espressioni booleane, dalle quali poi si ricaverà il circuito di decodifica.

mappe di karnaugh

Nelle mappe, indicheremo con **X** le *condizioni di indifferenza*.

Segmento a				
CD\AB	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	1	X	X
10	1	1	X	X

Segmento e				
CD\AB	00	01	11	10
00	1	0	X	1
01	0	0	X	0
11	0	0	X	X
10	1	1	X	X

Segmento b				
CD\AB	00	01	11	10
00	1	1	X	1
01	1	0	X	1
11	1	1	X	X
10	1	0	X	X

Segmento f				
CD\AB	00	01	11	10
00	1	1	X	1
01	0	1	X	1
11	0	0	X	X
10	0	1	X	X

Segmento c				
CD\AB	00	01	11	10
00	1	1	X	1
01	1	1	X	1
11	1	1	X	X
10	0	1	X	X

Segmento g				
CD\AB	00	01	11	10
00	0	1	X	1
01	0	1	X	1
11	1	0	X	X
10	1	1	X	X

Segmento d				
CD\AB	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	0	X	X
10	1	1	X	X

$$a = A + C + B \cdot D + \overline{B} \cdot \overline{D}$$
$$b = \overline{B} + \overline{C} \cdot \overline{D} + C \cdot D$$
$$c = A + B + \overline{C} + D$$
$$d = C + C \cdot \overline{D} + \overline{B} \cdot C + \overline{B} \cdot \overline{D}$$
$$e = \overline{B} \cdot \overline{D} + \overline{C} \cdot \overline{B}$$
$$f = (B + \overline{C}) \cdot (\overline{C} + \overline{D}) \cdot (A + B + \overline{D})$$
$$g = A + C \cdot \overline{D} + B \cdot \overline{C} + \overline{B} \cdot C$$

Alcune cose da sottolineare. Per i led **d** ed **f** si è preferito raggruppare i maxtermini (in realtà si è trattato di un esercizio, non di una necessità).

Queste scelte sono libere e si deducono osservando la mappa. Da un lato sono una cosa affascinante, perché il lavoro da compiere non è meccanico, ma lascia spazio anche alla

capacità ed alla fantasia del tecnico. Però questa è anche la morte stessa delle mappe di Karnaugh, perché per circuiti troppo complessi risulteranno improponibili per un essere umano.

la lista della spesa

Quanti componenti serviranno per realizzare il convertitore? Con un poco d'astuzia, meno di quel che sembra. I termini negati possono essere riutilizzati più volte (utilizzando solo 4 porte NOT, anziché 25), come ricavato dalle mappe di Karnaugh.

La negazione degli ingressi si esegue una volta soltanto e la si avrà a disposizione sino alla fine[8].

Si ottiene:

	AND	OR	NOT
a	2	3	-
b	2	2	-
c	-	3	-
d	3	3	-
e	2	1	-
f	2	4	-
g	3	3	-

In definitiva occorrono: 14 porte AND, 19 porte OR e le 4 porte NOT usate in precedenza.

teoremi di de morgan

I teoremi di De Morgan[9], o leggi di De Morgan, sono relativi alla logica booleana. Sono utilizzati per l'analisi di circuiti logici (elettrici, elettronici, pneumatici, comunque digitali, cioè ON-OFF).

I due teoremi sono duali, il primo è:

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

e si legge dicendo che se un elemento non appartiene ad $A \cdot B$ allora o non appartiene ad A, o non appartiene a B o non appartiene ad entrambi.

Il secondo teorema è il seguente:

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

Il secondo teorema si enuncia dicendo: se un elemento non appartiene ad A+B, allora non appartiene ad A e non appartiene a B.

È importante sottolineare che questi due teoremi possono essere estesi ad un numero elevato a piacere di variabili, vale a dire:

$$\overline{A \cdot B \cdot C \cdot D \cdot \ldots} = \overline{A} + \overline{B} + \overline{C} + \overline{D} + \ldots$$
$$\overline{A + B + C + D + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot \ldots$$

Il miglior metodo per dimostrare questi teoremi è non fidarsi e ricorrere alle tabelle della verità. Per il primo teorema si ha:

8 Bisogna fare attenzione ad un parametro (tipicamente fornito dal costruttore), il *fan-out*: il numero massimo di ingressi pilotabile da un'uscita di una porta logica.

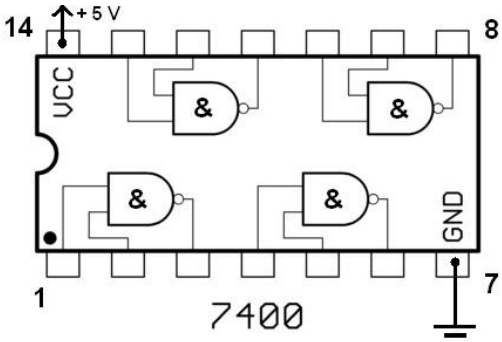
9 Tali teoremi prendono il nome dal matematico e logico britannico Augustus De Morgan.

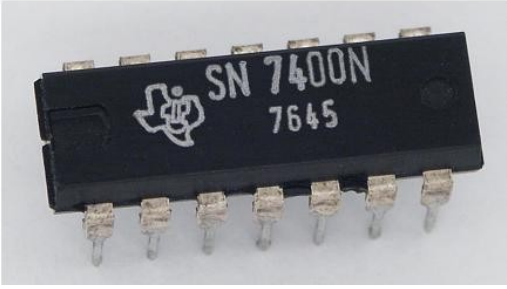
A	B	$\overline{A}$	$\overline{B}$	$A+B$	$\overline{A+B}$	$\overline{A}\cdot\overline{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Nel secondo caso si ottiene invece:

A	B	$\overline{A}$	$\overline{B}$	$A\cdot B$	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

transistor-transistor logic





La Transistor-Transistor Logic (TTL) è una classe di circuiti digitali integrati costruiti con l'utilizzo di transistor a giunzione bipolare (BJT) e resistori. È chiamata "logica transistor-transistor" perché (a differenza delle tecnologie precedenti, RTL e DTL), le funzioni delle resistenze e diodi sono svolte da transistor, come la funzione di porta logica (ad esempio OR) sia quella di amplificazione. È stata la prima tecnologia di circuiti integrati (IC) ad essere diffusa su scala globale in una grande varietà di applicazioni, come i computer, i controlli industriali, la strumentazione di laboratorio, nell'elettronica di consumo, le apparecchiature musicali, eccetera.

La tecnologia TTL si è diffusa a tal punto che si parla di circuiti "TTL-compatibili" per intendere circuiti di tecnologie successive (ad esempio CMOS) che mantengono la compatibilità con i livelli di tensione e di margine di rumore dei circuiti TTL. Tali circuiti sono usati come interfaccia per collegare tra loro stadi costruiti con logiche diverse (per esempio, per collegare un circuito TTL all'uscita di un CMOS, si metterà uno stadio intermedio con uscita "TTL-compatibile").

La Serie 7400 è un'importante famiglia di circuiti integrati realizzati in tecnologia TTL. Il primo della serie, l'integrato digitale 7400[10], contiene 4 porte logiche NAND, possiede 14 piedini (pin) due dei quali sono uno collegato all'alimentazione +5 V_{CC} (pin 14) e uno collegato a massa (GND, pin 7) per poter alimentare i transistor che formano il circuito delle porte logiche.

La serie è caratterizzata da una codifica standardizzata:

- logo del costruttore (nella figura a destra: Texas Instruments);
- una o più lettere che identificano il costruttore (nella figura "SN");
- il numero 74, tipico della famiglia logica;
- un numero (00), in altri casi costituito da più di due cifre, indicante le funzioni logiche svolte dal dispositivo.

10 Si legge "74 zero zero".