PROPRIETÀ E SINTESI

Proseguiamo il nostro percorso nell'elettronica digitale scoprendo alcune delle sue regole di base ed eseguendo la sintesi e la realizzazione di un semplice circuito logico a partire dalla sua tabella di verità.

Nel fascicolo 20 abbiamo presentato le porte logiche, che costituiscono le fondamenta dei circuiti logici combinatori. Abbiamo anche visto come a ognuna di esse sia associata 'un'espressione matematica'. Un circuito combinatorio, quindi, può essere rappresentato, oltre che in forma 'elettronica', anche attraverso una o più equazioni matematiche. Proprio come le espressioni matematiche, anche quelle logiche sono vincolate da una serie di regole e proprietà che permettono di interpretarle e di 'svolgere' correttamente i calcoli. Vediamo le principali, prima di dedicarci a un semplice esempio di sintesi logica.

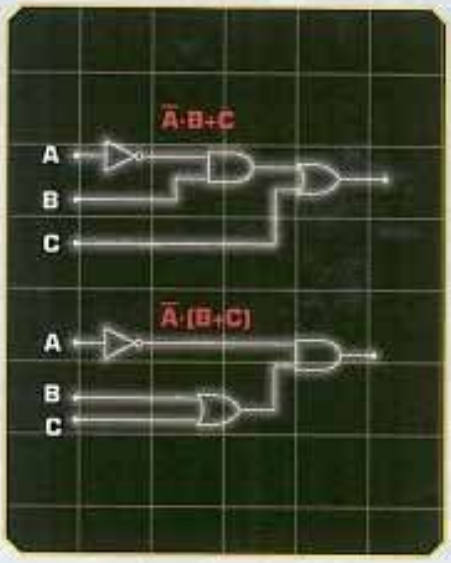
PRIORITÀ DEGLI OPERATORI›››

La prima regola necessaria per 'interpretare' correttamente un'espressione logica è stabilire le priorità per gli operatori (ossia stabilire in che ordine devono essere letti ed eseguiti i singoli operatori presenti). In particolare, vengono svolte prima le negazioni, poi i prodotti e per ultime le somme logiche. Per fare un esempio, consideriamo l'espressione seguente:

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

Basandoci sulle regole precedenti, il suo svolgimento avviene secondo questa sequenza di operazioni:

-negare il valore di A
-calcolare il prodotto A·B
-calcolare infine la somma $(\overline{A} \cdot B) + C$



Nel caso in cui si volessero alterare le priorità è possibile far ricorso all'uso delle parentesi. Ad esempio, se riscrivessimo l'espressione precedente in questa maniera:

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

la priorità di svolgimento verrebbe modificata (sarebbero eseguite prima le parentesi). La procedura di svolgimento sarebbe allora:

-negare il valore di A
-svolgere la somma (B+C)
-calcolare il prodotto $\overline{A} \cdot (B+C)$

Per comprendere come le due espressioni siano significativamente differenti, è possibile confrontare i due schemi elettrici equivalenti mostrati nella figura sopra, nella quale è visibile in maniera molto chiara il differente collegamento delle porte utilizzate.

PROPRIETÀ ASSOCIATIVA›››
Vale sia per le somme logiche che per i prodotti e asserisce che la somma (o il prodotto) di due o più variabili non cambia se a una o più di esse si sostituisce la somma (o il prodotto) delle stesse. Ad esempio:

$A+B+C=A+(B+C)=(A+B)+C=...$
$A\cdot B\cdot C=A\cdot(B\cdot C)=(A\cdot B)\cdot C=...$

PROPRIETÀ DISTRIBUTIVA›››
Consente di 'raccogliere' una variabile (o un'espressione) logica che compare in tutti gli addendi di una somma di prodotti. Ad esempio:

$A\cdot B+A\cdot C=A\cdot(B+C)$

PROPRIETÀ COMMUTATIVA›››
Vale sia per la somma che per il prodotto e afferma che invertendo addendi o fattori il risultato non cambia. Ad esempio:

$A\cdot B=B\cdot A$
$A+B=B+A$

PROPRIETÀ DI IDEMPOTENZA›››
Vale sia per la somma che per il prodotto e afferma che se si somma o si moltiplica più volte una variabile con se stessa, il risultato è uguale al valore della variabile. Ossia:

$(A+A+A+A+...)=A$
$(A\cdot A\cdot A\cdot A\cdot...)=A$

PROPRIETÀ DELLA NEGAZIONE›››
Vale la proprietà della doppia negazione, la quale afferma che se una variabile (o un'espressione) viene negata due volte, le due negazioni si annullano. Matematicamente:

$\overline{\overline{A}}=A$

TEOREMI DI DE MORGAN›››
Sono due importanti teoremi che consentono di riscrivere somme sotto forma di prodotti e viceversa. Tali teoremi affermano che:

$\overline{(A\cdot B\cdot C\cdot..N)}=\overline{A}+\overline{B}+\overline{C}+..\overline{N}$
$\overline{(A+B+C+..N)}=\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot..\overline{N}$

PROPRIETÀ DEL PRODOTTO›››
Valgono le seguenti proprietà (utili per 'semplificare' circuiti ed espressioni):

$A\cdot 0=0$
$A\cdot 1=A$
$A\cdot\overline{A}=0$

PROPRIETÀ DELLA SOMMA›››
Valgono le seguenti proprietà (utili per 'semplificare' circuiti ed espressioni):

$A+0=A$
$A+1=1$
$A+\overline{A}=1$

UN SEMPLICE ESEMPIO DI SINTESI›››

Ora che abbiamo elencato le principali proprietà degli operatori logici, cerchiamo di capire cosa significa 'progettare' un circuito logico (di tipo combinatorio) e in che modo è possibile farlo. Progettare un circuito combinatorio significa ideare una rete logica in grado di elaborare i dati in ingresso secondo lo schema desiderato.

Il processo di progettazione inizia con l'individuazione delle variabili di ingresso e di uscita di cui abbiamo bisogno. Una volta identificate le variabili logiche coinvolte si procede alla compilazione di

simbolo da trasmettere (3 ingressi)

bit di parità (1 uscita)

Lo schema del generatore di parità che progetteremo, con tre variabili di ingresso (X_n) e una di uscita (Par).

una tabella di verità che descrive in modo completo il comportamento della rete logica (ingressi e uscite). In ultimo si 'sintetizza' il circuito appena definito (ossia si ricava dalla tabella di verità l'espressione logica equivalente). Per capire in modo 'interattivo' l'intero processo di progettazione, ricorriamo a un semplice esempio concreto: **progettiamo un piccolo generatore di parità per simboli a 3 bit.**

IL GENERATORE DI PARITÀ ›››

La generazione dei bit di parità è una semplicissima tecnica di **rivelazione degli errori** utilizzata nella trasmissione digitale dei dati. Il suo funzionamento è estremamente semplice: in pratica, a ogni simbolo trasmesso (un simbolo è inteso come un 'insieme di bit', nel nostro esempio 3) viene associato un ulteriore bit chiamato **'bit di parità'**. Esso ha la funzione di **'rendere pari' il numero di bit uguali a 1 del simbolo trasmesso.** Ciò permette di poter 'contare' i bit 'alti' ('1') durante il processo di ricezione e di rilevare, in questa maniera, eventuali errori di trasmissione (se viene ricevuto un numero dispari di '1' significa che vi è stato un errore nella trasmissione). Ricorrendo a un esempio, se viene trasmesso il simbolo '101', il bit di parità associato sarà uguale a '0'. Al contrario, se il simbolo in trasmissione è '010', il bit di parità vale '1', poiché nella sequenza di bit emessi vi è un numero dispari di '1'. Descritto il **funzionamento** alla base del circuito, è necessario, come detto in precedenza, **identificare le variabili**: avremo **tre variabili in ingresso** (i tre bit del simbolo) e **una in uscita** (il bit di parità). **Compiliamo ora la tabella di verità** assegnando a 'Par' il valore '1' in tutte le combinazioni di ingresso che presentano un numero dispari di bit 'alti'. Ecco la tabella di verità corrispondente al circuito desiderato:

X_3	X_2	X_1	Par
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Procediamo ora alla **sintesi** vera e propria del circuito. Una delle tecniche più diffuse per sintetizzare le tabelle di verità consiste nella **scrittura di una somma di prodotti tra le combinazioni di ingresso che portano a '1' l'uscita**. Vediamo come procedere. **Iniziamo identificando tutte le righe con uscita alta** (evidenziate in rosso nella tabella di verità). Il comportamento di ognuna di esse può essere replicato per mezzo di una **porta logica 'AND' a 'n' entrate** (nel nostro caso 3), **che svolga il prodotto tra le variabili di ingresso, lasciate invariate se uguali a 1 o negate se, invece, uguali a 0.** La seconda riga della tabella, che ha ingressi X_3=0, X_2=0, X_1=1, sarà rappresentata dal prodotto:

$$\overline{X_3} \cdot \overline{X_2} \cdot X_1$$

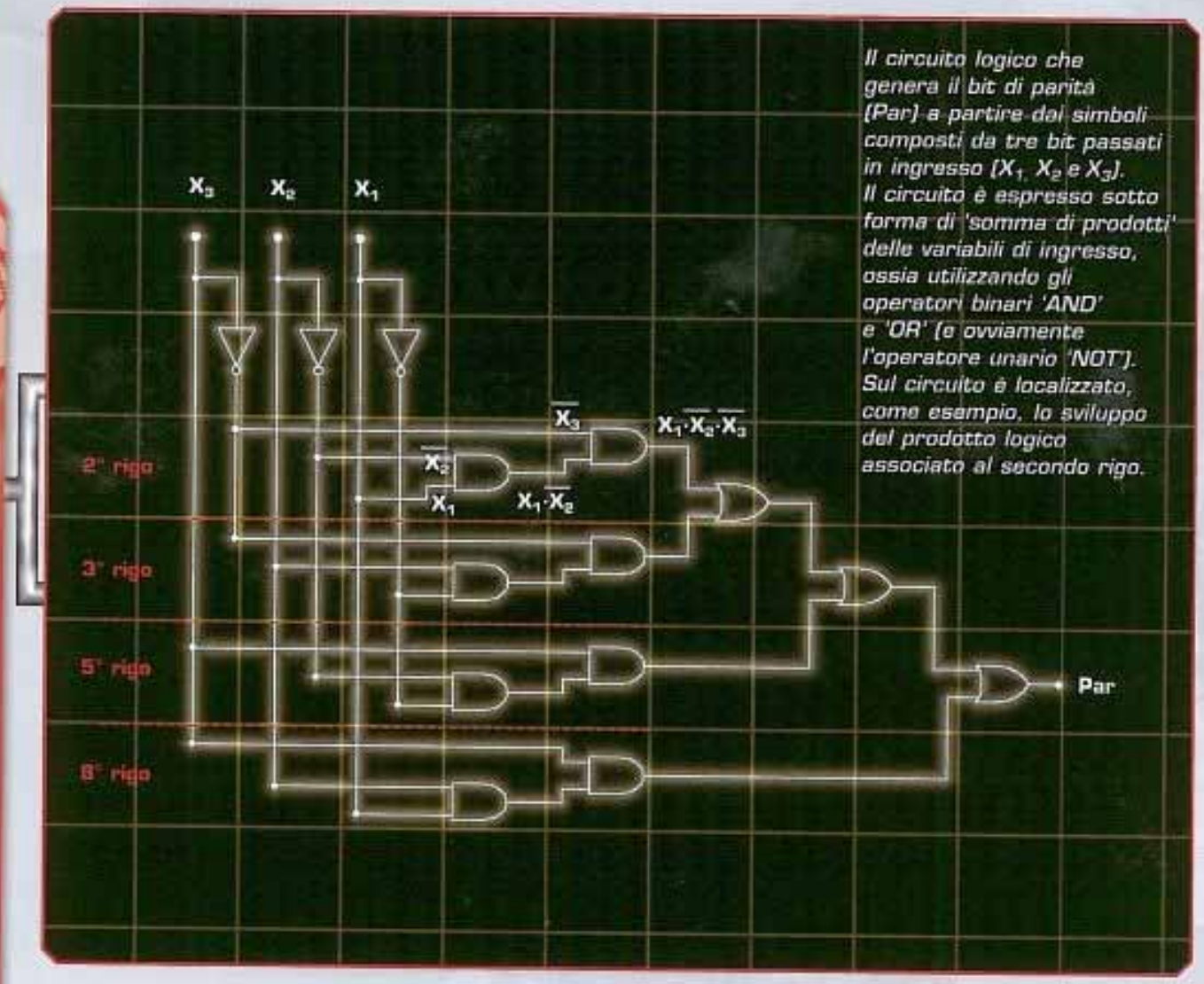
[X_3 e X_2 sono negate in quanto uguali a '0', mentre X_1 rimane invariata dato che è pari a '1']. In ugual modo, la terza riga corrisponderà al prodotto:

$$\overline{X_3} \cdot X_2 \cdot \overline{X_1} \qquad (X_1=X_3=0,\ X_2=1)$$

e così via per le due rimanenti righe con Par=1'. **Definiti tutti i casi 'alti' è sufficiente 'sommare' (operatore 'OR') le sottoespressioni ottenute per avere il circuito logico finale.** L'espressione completa del circuito sarà quindi:

$$\begin{aligned} Par = {} & \overline{X_3} \cdot \overline{X_2} \cdot X_1 + \quad \text{2° riga} \\ & + \overline{X_3} \cdot X_2 \cdot \overline{X_1} + \quad \text{3° riga} \\ & + X_3 \cdot \overline{X_2} \cdot \overline{X_1} + \quad \text{5° riga} \\ & + X_3 \cdot X_2 \cdot X_1 \quad \text{8° riga} \end{aligned}$$

Tale espressione è equivalente al circuito logico combinatorio



Il circuito logico che genera il bit di parità (Par) a partire dai simboli composti da tre bit passati in ingresso (X_1, X_2 e X_3). Il circuito è espresso sotto forma di 'somma di prodotti' delle variabili di ingresso, ossia utilizzando gli operatori binari 'AND' e 'OR' (e ovviamente l'operatore unario 'NOT'). Sul circuito è localizzato, come esempio, lo sviluppo del prodotto logico associato al secondo rigo.

mostrato qui sopra in questa pagina. Ora dedichiamoci ad alcune brevi considerazioni. Innanzitutto si nota che nel circuito **non sono state usate porte logiche 'AND' e 'OR' a più di due ingressi**. Questo proprio perché normalmente le porte logiche integrate rispecchiano la natura binaria dell'operatore booleano corrispondente (sono quindi a due ingressi). Esistono comunque porte complesse (ad esempio AND a quattro ingressi), ma noi faremo riferimento alle porte a due sole entrate, che sono un buon indice della '**complessità**' del circuito progettato.

Uno dei modi per 'quantificare' la complessità di una rete logica, infatti, consiste nel 'contare' fisicamente il numero di porte elementari richieste per realizzarla. Nel caso del sistema appena progettato è necessario utilizzare tre porte 'NOT', otto 'AND' e tre 'OR', per un totale di **14 porte logiche** (come puoi contare nello schema). Il **numero di porte logiche** utilizzate è importante in quanto è direttamente legato al **numero di dispositivi elettronici richiesti** per realizzare il circuito. Poiché con l'aumentare di questo numero aumentano anche i **costi** e i **consumi**, l'obiettivo è sempre quello di cercare di **mantenere il sistema il più semplice possibile**. In questo caso, poiché in ognuno dei componenti usati nel precedente StepbyStep erano contenute sei porte 'NOT', quattro 'AND' o quattro 'OR', per poter costruire il nostro generatore di parità avremo bisogno di **quattro differenti circuiti integrati** (uno per le porte 'NOT', uno per le porte 'OR' e due per le 'AND'). Ma i circuiti logici possono anche essere semplificati...

STEPbySTEP

UN GENERATORE DI PARITÀ

In questo StepbyStep sperimenterai il generatore di parità che abbiamo progettato all'interno dell'articolo. Prima di passare alla realizzazione 'pratica' del circuito, tuttavia, soffermiamoci sull'espressione logica ricavata e sullo schema elettronico a essa associato, mostrato nella pagina precedente. Come abbiamo già avuto modo di osservare, l'espressione ricavata richiede per la sua implementazione fisica l'utilizzo di ben 14 porte differenti (tre porte 'NOT', otto 'AND' e tre 'OR'). Vediamo ora in che modo è possibile semplificarla. Innanzitutto richiamiamo l'espressione logica alla base del circuito:

$$Par = \overline{X_3} \cdot \overline{X_2} \cdot X_1 + \overline{X_3} \cdot X_2 \cdot \overline{X_1} + X_3 \cdot \overline{X_2} \cdot \overline{X_1} + X_3 \cdot X_2 \cdot X_1$$

Da essa puoi vedere come nei primi addendi compaia il fattore NOT X_3, mentre negli ultimi due sia comune il termine X_3. Raccogliamo questi due termini sfruttando la proprietà distributiva. L'equazione diventa:

$$Par = \overline{X_3} \cdot \left(\overline{X_2} \cdot X_1 + X_2 \cdot \overline{X_1}\right) + X_3 \cdot \left(\overline{X_2} \cdot \overline{X_1} + X_2 \cdot X_1\right)$$

Ora osserva il contenuto delle parentesi. La prima delle due è un'espressione equivalente alla scrittura:

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

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

La tabella di verità associata a tale espressione è mostrata qui sopra e, come puoi osservare, è identica a quella operatore logico XOR, presentata nel fascicolo 20. Possiamo quindi sostituire l'intera sottoespressione con l'espressione equivalente X_2 XOR X_1. Alla stessa maniera osserva il secondo blocco: anch'esso ha un comportamento già noto. Assume, infatti, valore di uscita alto solo quando le variabili X_1 e X_2 sono uguali: in pratica è un'espressione equivalente all'operatore logico XNOR. Possiamo sostituire anch'esso con la corrispondente forma semplificata. La rete logica diviene quindi:

$$Par = \overline{X_3} \cdot (X_2 \oplus X_1) + X_3 \cdot \overline{(X_2 \oplus X_1)}$$

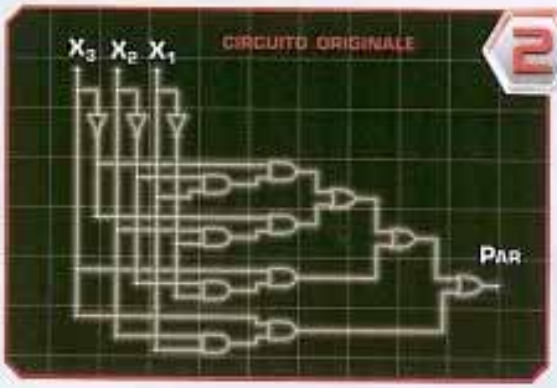
Ora soffermati ancora sull'attuale struttura dell'espressione logica: puoi osservare nuovamente una forma del tipo $\overline{A} \cdot B + A \cdot \overline{B}$ (con $B=(X_2$ XOR $X_1)$ e $A=X_3$). Come abbiamo fatto in precedenza possiamo semplificare l'attuale espressione per mezzo dell'operatore logico XOR. La rete logica diviene allora:

$$Par = X_3 \oplus (X_2 \oplus X_1)$$

Abbiamo ridotto il circuito iniziale, composto da ben 14 porte logiche, a un circuito equivalente formato da due sole porte elementari. Con un po' di pratica, è possibile applicare le proprietà degli operatori logici per ridurre fortemente la complessità dei circuiti logici progettati. Proveremo a questo punto a realizzare il circuito semplificato e a verificare la sua equivalenza con quello originale. Le porte XOR che userai sono contenute nell'integrato 74HC86.

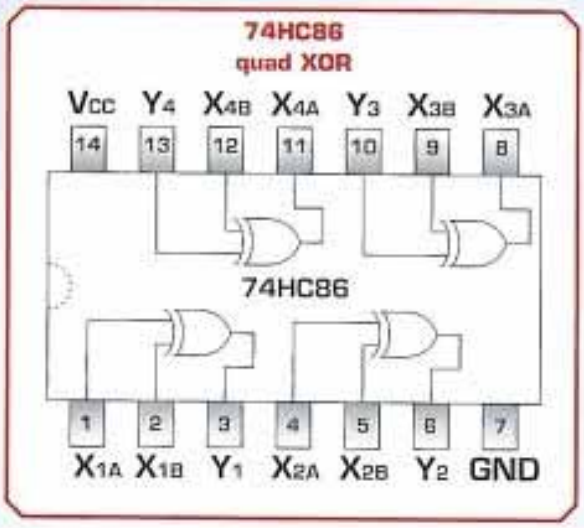
Ecco il materiale necessario per questo esperimento:

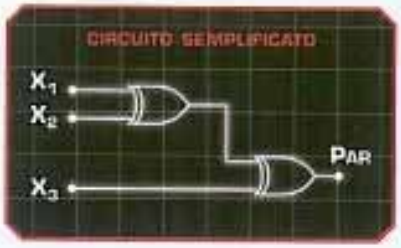
‹1› una breadboard
‹2› fili per breadboard
‹3› un 74HC86 (IC_{XOR})
‹4› un microswitch a 4 vie (SW)
‹5› tre LED rossi (D_1, D_2, D_3)
‹6› tre resistori da 4,7 kohm (R_H)
‹7› un LED verde (D_{PAR})
‹8› quattro resistori da 220 ohm (R_D)

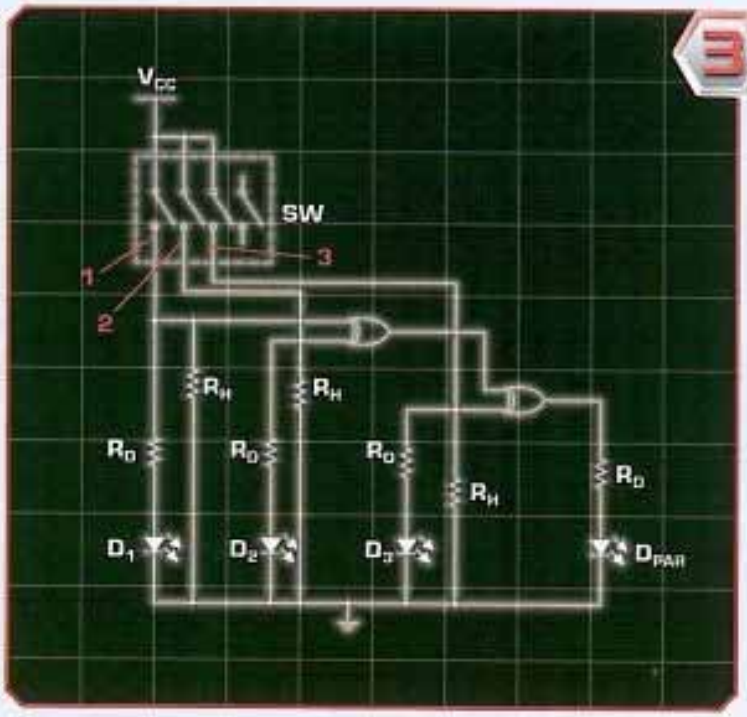


2

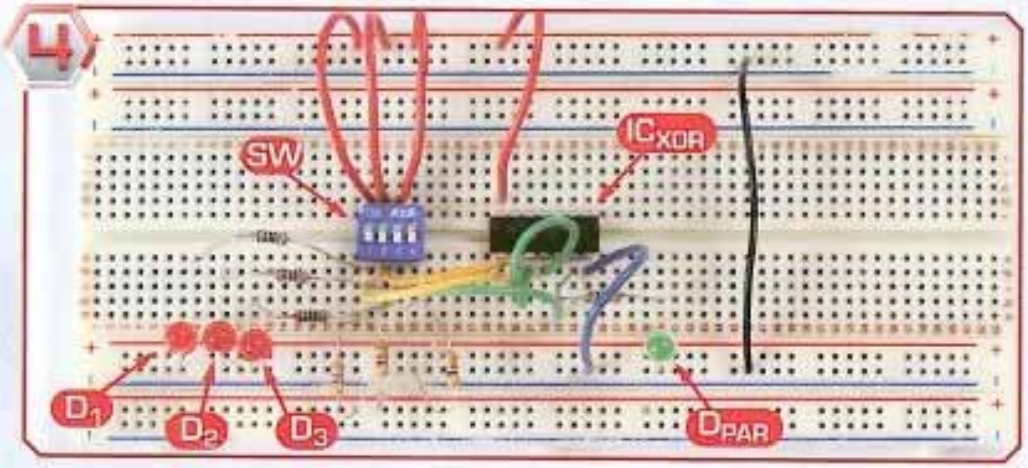
In questo StepbyStep realizzerai il circuito generatore di parità che abbiamo progettato e 'semplificato' nelle precedenti pagine. Come hai potuto vedere, infatti, applicando opportunamente le proprietà dell'algebra booleana e l'equivalenza delle espressioni logiche, siamo riusciti a ridurre il circuito originale, composto da ben 14 porte logiche (figura a lato), ricavandone uno equivalente formato da porte XOR (figura sotto). Tali porte elementari sono contenute all'interno del circuito integrato 74HC86 (della stessa sottofamiglia di quelli usati nel precedente Workshop), la cui struttura interna è mostrata nello schema a sinistra. Come tutti gli integrati logici, anche questo dovrà essere alimentato attraverso i pin 14 e 7.



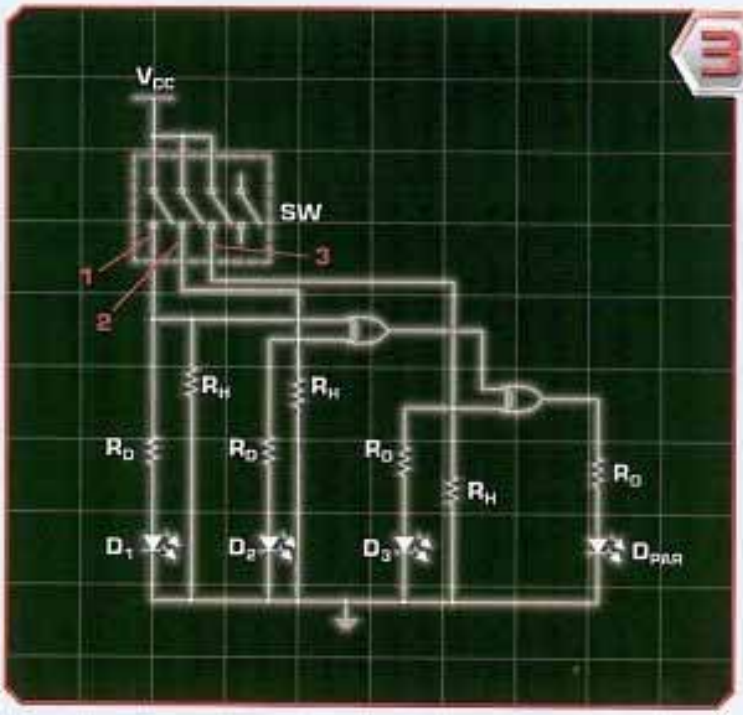




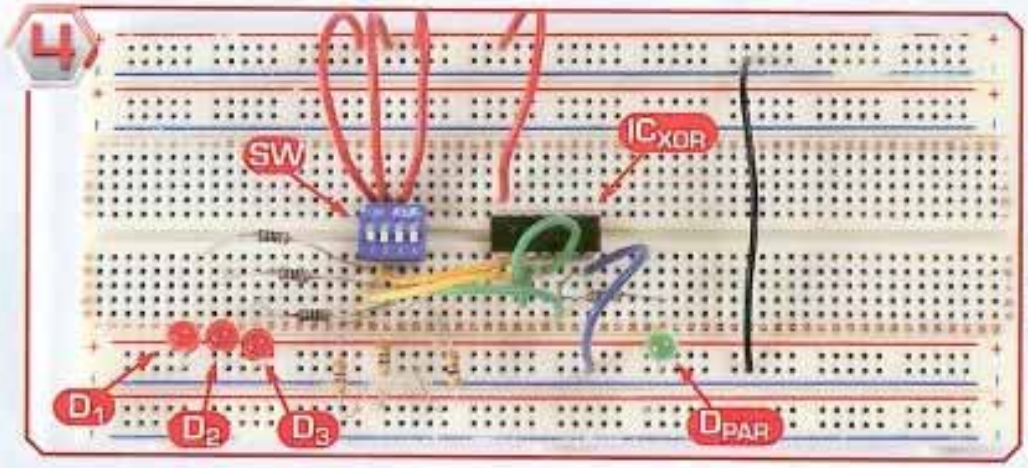
A lato puoi vedere mostrato il circuito che dovrai realizzare. Come al solito andrà alimentato per mezzo di un portapile da quattro pile 'stilo', i cui fili saranno associati al terminale V_{CC} e alla massa del circuito. Ricordati che a tali terminali dovranno essere collegati anche i pin 14 (V_{CC}) e 7 (GND) del circuito integrato 74HC86. I primi tre interruttori del microswitch (SW) ti consentiranno di impostare il valore delle tre linee di ingresso al circuito. Portandoli su 'ON', infatti, ti sarà possibile collegare gli ingressi a V_{CC}, e quindi al valore logico '1'. Aprendoli, invece, (ossia portandoli su 'OFF'), i pin di ingresso delle porte logiche saranno collegati a massa attraverso i resistori R_H. R_H svolge un ruolo importante, in quanto evita il verificarsi di cortocircuiti nel momento in cui vengono chiusi gli interruttori di SW (di fatto limita il flusso di corrente tra V_{CC} e massa).



In figura puoi vedere il circuito montato su breadboard. I tre LED in basso a sinistra, indicati con D_1, D_2 e D_3 sono i tre dispositivi luminosi che ti permettono di visualizzare lo stato delle linee di ingresso. Si accenderanno in presenza di ingressi alti (interruttori su ON) e si spegneranno in caso di ingressi bassi. Un comportamento analogo vale per il LED D_{PAR}, che ti mostrerà il valore del bit di parità generato dal circuito logico che abbiamo progettato. Se avrai montato correttamente il circuito, dopo averlo alimentato scoprirai che indipendentemente dallo stato degli interruttori del microswitch (e quindi delle variabili logiche di ingresso) il numero totale dei LED accesi sarà sempre pari (ossia 0, 2 o 4). Se vuoi, per fare pratica, puoi riprogettare il circuito in modo da farlo funzionare con quattro linee di ingresso (la tabella di verità sarà composta, quindi, da 2^4=16 righe).



3 A lato puoi vedere mostrato il circuito che dovrai realizzare. Come al solito andrà alimentato per mezzo di un portapile da quattro pile 'stilo', i cui fili saranno associati al terminale V_{CC} e alla massa del circuito. Ricordati che a tali terminali dovranno essere collegati anche i pin 14 (V_{CC}) e 7 (GND) del circuito integrato 74HC86. I primi tre interruttori del microswitch (SW) ti consentiranno di impostare il valore delle tre linee di ingresso al circuito. Portandoli su 'ON', infatti, ti sarà possibile collegare gli ingressi a V_{CC}, e quindi al valore logico '1'. Aprendoli, invece, (ossia portandoli su 'OFF'), i pin di ingresso delle porte logiche saranno collegati a massa attraverso i resistori R_H. R_H svolge un ruolo importante, in quanto evita il verificarsi di cortocircuiti nel momento in cui vengono chiusi gli interruttori di SW (di fatto limita il flusso di corrente tra V_{CC} e massa).



4 In figura puoi vedere il circuito montato su breadboard. I tre LED in basso a sinistra, indicati con D_1, D_2 e D_3 sono i tre dispositivi luminosi che ti permettono di visualizzare lo stato delle linee di ingresso. Si accenderanno in presenza di ingressi alti (interruttori su ON) e si spegneranno in caso di ingressi bassi. Un comportamento analogo vale per il LED D_{PAR}, che ti mostrerà il valore del bit di parità generato dal circuito logico che abbiamo progettato. Se avrai montato correttamente il circuito, dopo averlo alimentato scoprirai che indipendentemente dallo stato degli interruttori del microswitch (e quindi delle variabili logiche di ingresso) il numero totale dei LED accesi sarà sempre pari (ossia 0, 2 o 4). Se vuoi, per fare pratica, puoi riprogettare il circuito in modo da farlo funzionare con quattro linee di ingresso (la tabella di verità sarà composta, quindi, da 2^4=16 righe).