Reti Logiche
Appello del 19 marzo 2007 – Seconde prove

(D2) Si consideri la funzione di commutazione $f_k(x_1, x_2, \ldots, x_n)$ che vale 1 se e solo se esattamente *k* variabili sono uguali ad 1. Quanti mintermini e quanti implicanti primi ha tale funzione?

◆ ◆ ◆ ◆ ◆ ◆

Posto ad esempio $n = 4$ e $k = 2$, i mintermini della funzione $f_2(x_1, x_2, x_3, x_4)$ saranno 6:

$$x_1 x_2 \overline{x}_3 \overline{x}_4$$
$$x_1 \overline{x}_2 x_3 \overline{x}_4$$
$$x_1 \overline{x}_2 \overline{x}_3 x_4$$
$$\overline{x}_1 x_2 x_3 \overline{x}_4$$
$$\overline{x}_1 x_2 \overline{x}_3 x_4$$
$$\overline{x}_1 \overline{x}_2 x_3 x_4$$

Nel caso generale, il numero di mintermini sarà pari al numero $C_{n,k}$ delle *combinazioni* (senza ripetizioni) di *n* oggetti presi *k* a *k*, ossia

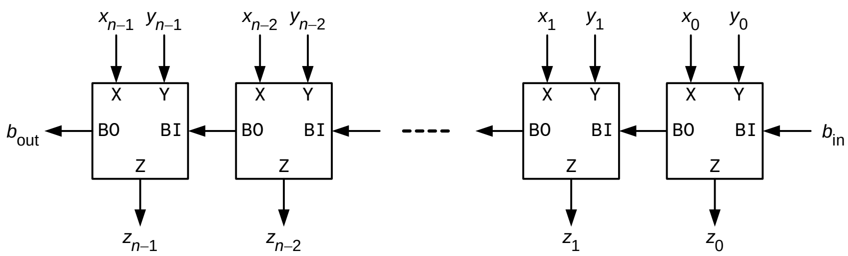
$$C_{n,k} = \binom{n}{k} = \frac{n!}{k!(n-k)!}$$

È possibile ricavare un implicante da una coppia di mintermini o di implicanti solo se la coppia differisce per una sola variabile, che appare diretta in un elemento della coppia e negata nell'altro, mentre tutti gli altri letterali rimangono immutati. Nel caso in questione, in una qualunque coppia di mintermini *A*, *B* vi saranno *almeno due* letterali che presentano una variazione (almeno una variabile diretta in *A* deve diventare complementata in *B*, mentre almeno una variabile complementata in *A* deve diventare diretta in *B*, in maniera da mantenere costante il numero totale di variabili in forma diretta), e dunque non è possibile assemblare termini prodotto con meno di *n* variabili. Inoltre, nei casi degeneri in cui $k = n$ si ha un solo mintermine e dunque un solo implicante primo. Di conseguenza il numero di implicanti primi è sempre uguale al numero di mintermini.

(D3) Definire la struttura di una rete iterativa per la *sottrazione aritmetica* di due numeri binari, e sviluppare il progetto della cella elementare. (*Nota:* Non è ammesso l'uso di addizionatori.)

◆◆◆◆◆◆

La rete iterativa per la sottrazione aritmetica di due numeri binari ha la stessa struttura di quella per l'addizione aritmetica, salvo la diversa struttura della cella e la diversa interpretazione del segnale che si propaga da una cella all'altra:



dove $X = x_{n-1}x_{n-2} \ldots x_1x_0$ e $Y = y_{n-1}y_{n-2} \ldots y_1y_0$ sono i due operandi, detti ripettivamente *minuendo* e *sottraendo*, mentre $Z = z_{n-1}z_{n-2} \ldots z_1z_0$ è la *differenza*.

In analogia con la normale sottrazione decimale, la cella generica deve realizzare la funzione

$$z_k = x_k - y_k - b_k$$

(dove il simbolo – deve essere inteso come *sottrazione aritmetica*), essendo b_k il *prestito in ingresso* (*borrow-in*) BI. Ancora in analogia con la normale sottrazione decimale, la cella deve anche generare un *prestito in uscita* (*borrow-out*) b_{k+1} sul terminale BO quando $x_k < y_k + b_k$, in modo che al minuendo possa essere sommata la base di numerazione (10 nel caso decimale, 2 nel caso binario) rendendo così il risultato z_k non minore di zero.

Con queste premesse, si costruisce la tavola di verità della cella:

x_k	y_k	b_k	b_k' ($x_k > y_k + b_k$)	operazione aritmetica	z_k
0	0	0	0	0 – 0 – 0 = 0	0
0	0	1	1	(2 + 0) – 0 – 1 = 1	1
0	1	0	1	(2 + 0) – 1 – 0 = 1	1
0	1	1	1	(2 + 0) – 1 – 1 = 0	0
1	0	0	0	1 – 0 – 0 = 1	1
1	0	1	0	1 – 0 – 1 = 0	0
1	1	0	0	1 – 1 – 0 = 0	0
1	1	1	1	(2 + 1) – 1 – 1 = 1	1

Le mappe di Karnaugh per le due funzioni z_k, b_k' sono:

$x_k y_k \backslash b_k$	0	1
00		**1**
01	**1**	
11		**1**
10	**1**	

z_k

$x_k y_k \backslash b_k$	0	1
00		**1**
01	**1**	**1**
11		**1**
10		

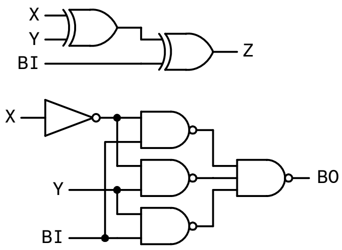
$b_k{}'$

e le rispettive funzioni sono:

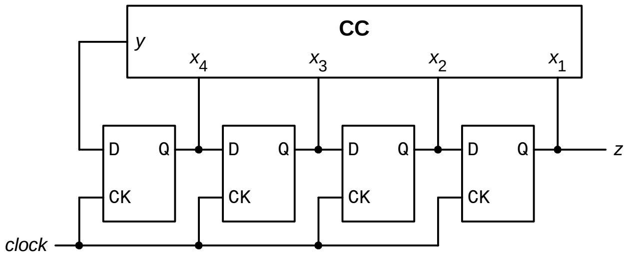
$$
\begin{aligned}
z_k &= \overline{x}_k \overline{y}_k b_k + \overline{x}_k y_k \overline{b_k} + x_k \overline{y}_k \overline{b_k} + x_k y_k b_k = \\
&= x_k \oplus y_k \oplus b_k
\end{aligned}
$$

$$b_k{}' = \overline{x}_k b_k + \overline{x}_k y_k + y_k b_k$$

Una possibile implementazione della cella è la seguente:



(D4) Il circuito illustrato nella figura a fianco deve generare all'uscita z la sequenza S = `11000101001`, che si ripete periodicamente e indefinitamente. Assumendo che lo stato iniziale sia tale che $x_1 = x_2 = 1$, $x_3 = x_4 = 0$, e che in tali condizioni su z sia presente il primo bit della sequenza S, determinare la funzione $y = f(x_1, x_2, x_3, x_4)$ che il circuito combinatorio CC deve realizzare, e ricavarne quindi un'implementazione.



◆ ◆ ◆ ◆ ◆ ◆

La tavola di verità del circuito combinatorio CC si costruisce facilmente osservando che, per ogni quaterna di elementi consecutivi della sequenza S, esso deve generare l'elemento successivo. Così, quando $x_1 x_2 x_3 x_4 = 1100$ (i primi quattro elementi di S) CC deve generare 0 (quinto elemento di S); quando $x_1 x_2 x_3 x_4 = 1000$ (elementi di S dal secondo al quinto) CC deve generare 1 (sesto elemento di S), e così via. Inoltre, affinché la sequenza generata sia periodica, quando ad $x_1 x_2 x_3 x_4$ vengono presentati gli ultimi 4 elementi di S, CC deve generare 1 (il primo elemento di S); quando ad $x_1 x_2 x_3$ vengono presentati gli ultimi tre elementi di S e ad x_4 il primo, CC deve generare 1 (secondo elemento di S), e così via. Le combinazioni di $x_1 x_2 x_3 x_4$ che non dovessero apparire mai come sottosequenze di S di lunghezza 4, genereranno condizioni non specificate (*don't care*).

In definitiva:

x_1	x_2	x_3	x_4	y
0	0	0	0	–
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	–
0	1	1	1	0

x_1	x_2	x_3	x_4	y
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	–
1	1	0	0	0
1	1	0	1	–
1	1	1	0	0
1	1	1	1	–

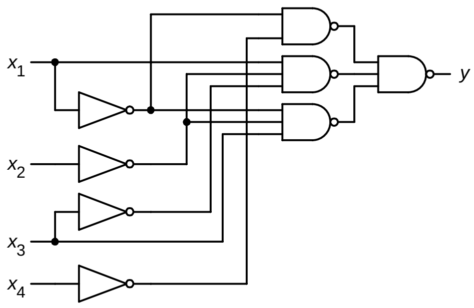
La mappa di Karnaugh corrispondente è:

x_3x_4 \ x_1x_2	00	01	11	10
00	–	1		1
01			–	1
11	1		–	–
10	1	–		

da cui deriva l'espressione

$$y = \overline{x}_1 \overline{x}_4 + x_1 \overline{x}_2 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3$$

e una possibile implementazione in NAND di NAND:


x_1
x_2
x_3
x_4
y