Interfaccia BINBCD

Un'interfaccia BINBCD è attestata su una linea seriale esterna `XIN`, sincronizzata con un clock `XCLK`, dalla quale riceve in continuazione blocchi da 32 bit consecutivi, in corrispondenza al primo dei quali è attivo un segnale esterno `XSYNC`. I dati contenuti nel blocco, interpretati come numeri binari assoluti di cui il primo bit è il più significativo, devono essere convertiti in base 10 con le cifre espresse in formato BCD. Tali cifre BCD devono infine essere emesse su un gruppo di 4 linee di uscita `XOUT[0-3]`, una alla volta, sincronizzate al clock `XCLK`, a partire dalla cifra più significativa diversa da 0 e a finire con la cifra meno significativa; per tutta la durata dell'emissione delle cifre deve inoltre essere attivo un segnale di uscita `XBCD`.

(*) Progetto d'esame per il corso di Reti Logiche, appello del 2006-12-18, Laurea Specialistica in Ingegneria Informatica, Università di Roma "La Sapienza".

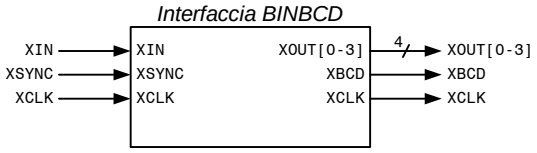


Fig. 1 L'interfaccia BINBCD accetta in ingresso tre segnali da un bit ciascuno (`XIN`: dati da convertire; `XSYNC`: sincronismo; `XCLK`: clock) ed emette in uscita un canale dati `XOUT[0-3]` da 4 bit, un segnale di sincronismo `XBCD` e un clock `XCLK` che coincide col clock in ingresso. L'interfaccia è di tipo *stand-alone*, ossia, non vi sono connessioni al bus di alcuna CPU.

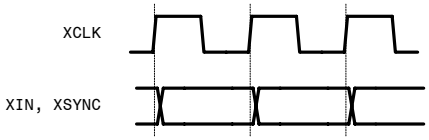


Fig. 2 Possiamo assumere che la relazione temporale tra `XCLK` e gli altri segnali in ingresso sia quella illustrata in figura, dove il segnale commuta in corrispondenza al fronte di salita di `XCLK` e rimane stabile per tutto il resto del periodo di clock.

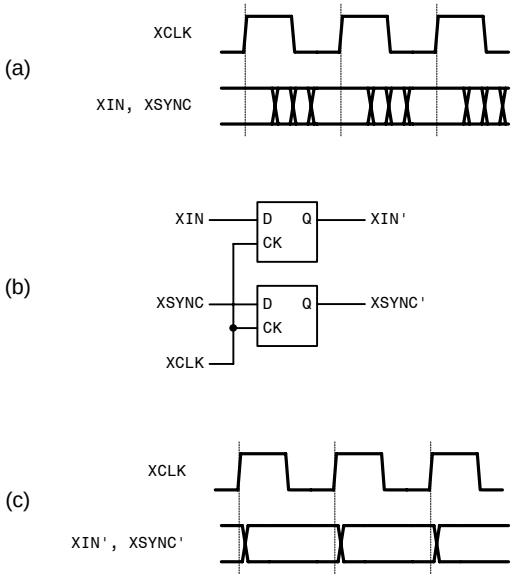


Fig. 3 Se così non fosse, se ad esempio i segnali di ingresso fossero stabili solo in un intorno del fronte attivo di `XCLK` (*a*), possiamo risincronizzare i segnali mediante flip-flop D (*b*), e ricondurci così (*c*) alla situazione di Fig. 2.

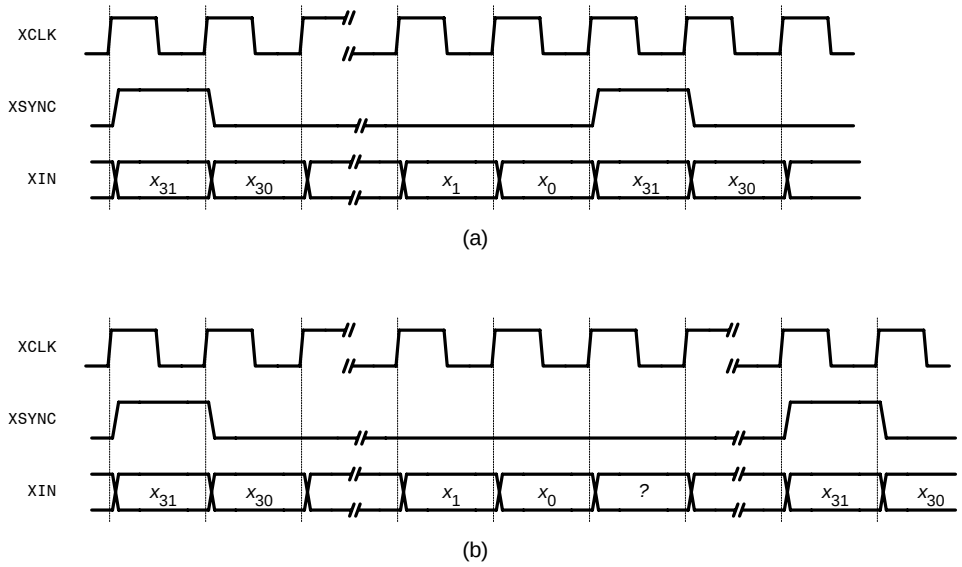


Fig. 4 Il segnale XSYNC identifica l'inizio di un *blocco* da 32 bit { x_{31} x_{30} ... x_1 x_0 } sulla linea XIN; il contenuto del blocco va interpretato come numero intero assoluto di cui x_{31} è il bit più significativo. Si noti come, in assenza di specifiche nel testo del problema, (*a*) un blocco può iniziare immediatamente dopo la fine del blocco precedente, oppure (*b*) possono essere presenti altri dati (che andranno ignorati) tra due blocchi successivi; l'interfaccia deve essere in grado di gestire entrambe le situazioni.

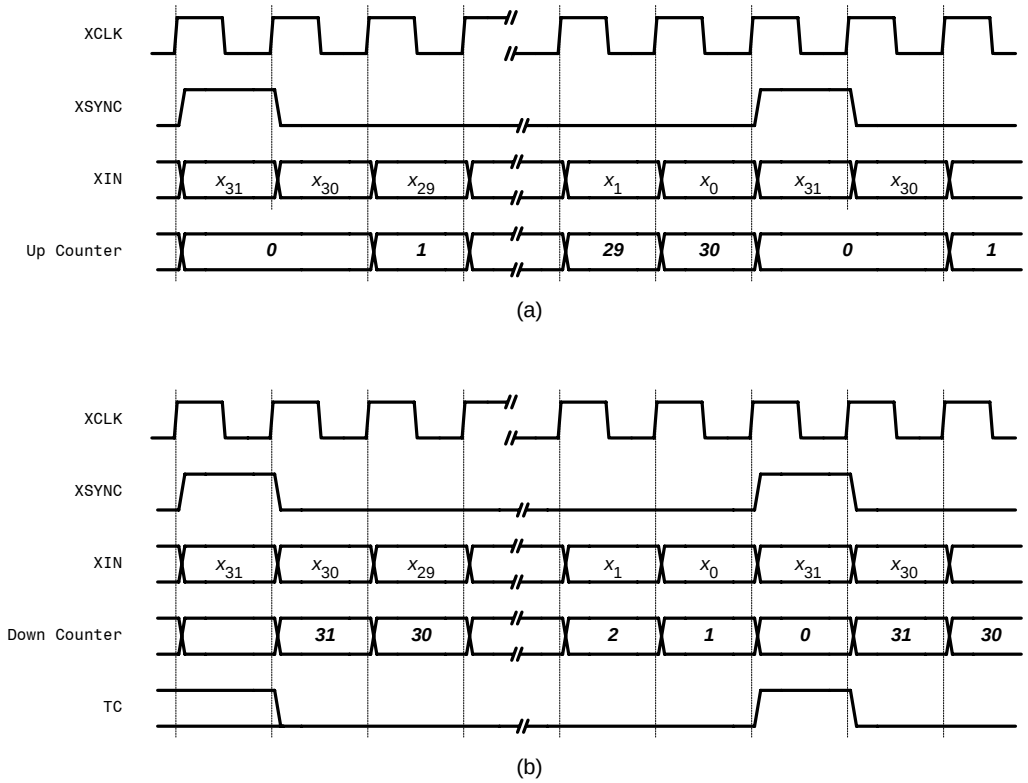
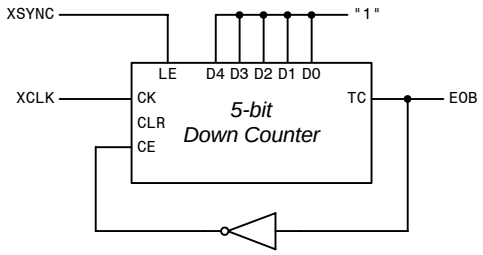
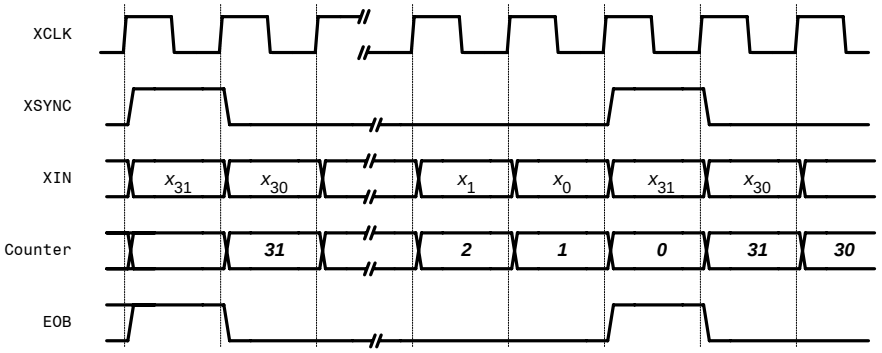


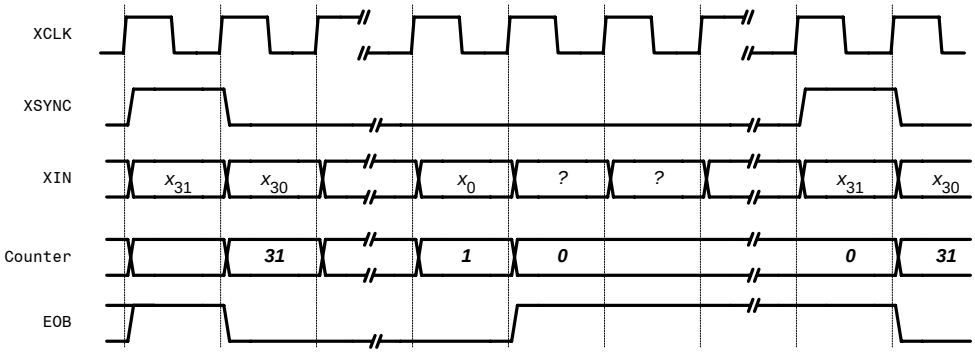
Fig. 5 Data la possibilità di blocchi non necessariamente contigui, XSYNC può essere utilizzato per identificare l'inizio di un blocco ma non la fine del blocco precedente; di conseguenza, è necessario contare i bit del blocco per poterne identificare la fine. Il segnale XSYNC inizializzerà un contatore mod 32, il cui Terminal Count potrà così identificare la fine del blocco. Tuttavia, il contatore non potrà essere inizializzato in modo asincrono, perché in tal caso il conteggio iniziale avrà durata di *due* clock, e nel caso di blocchi contigui il conteggio terminale non verrà raggiunto; la situazione è illustrata in (*a*) nel caso di un up-counter mod 32. Se invece l'inizializzazione è di tipo sincrono (come illustrato in (*b*) nel caso di un down-counter) anche l'arrivo immediato di un nuovo blocco consente al Terminal Count TC di diventare attivo.



(a)



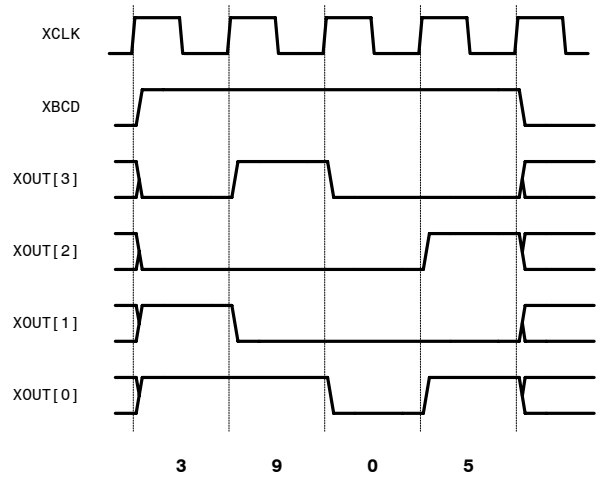
(b)



(c)

Fig. 6 Per il conteggio dei bit del blocco si utilizzerà dunque un Down Counter a 5 bit (*a*), inizializzato da XSYNC in modo sincrono col conteggio 31 (11111); il Terminal Count diventa così un indicatore di fine blocco (End of Block, EOB). Per evitare EOB multipli nel caso di blocchi molto distanziati tra loro, qualora il Counter venisse lasciato avanzare liberamente, è sufficiente disabilitare il conteggio una volta raggiunto TC. Le temporizzazioni in (*b*) e (*c*) rappresentano il comportamento del contatore nel caso di blocchi contigui e, rispettivamente, non contigui.

(a)



(b)

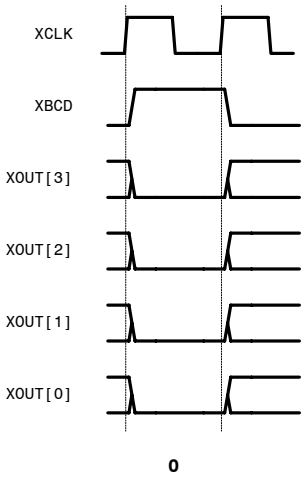


Fig. 7 Il numero intero rappresentato dai bit del blocco x_{31-0} va convertito in base 10, e le cifre risultanti, espresse in formato BCD, vanno emesse sulle linee XOUT[3-0], sincronizzandole con XCLK e identificando la loro presenza con un segnale XBCD. Se ad esempio la conversione del blocco dà come risultato il numero 3905, corrispondente alla sequenza BCD { 0011, 1001, 0000, 0101 }, le uscite si devono comportare come illustrato in (*a*); si noti come XBCD diventi attivo solo in corrispondenza delle cifre significative del numero in base 10 (con esclusione quindi di eventuali zeri non significativi), e come i dati emessi quando XBCD è inattivo possano essere qualsivogliano. Ovviamente, se la conversione dà come risultato 0, dovrà essere emessa (*b*) una sola cifra BCD pari a { 0000 } e XBCD resterà attivo per un solo periodo di XCLK.

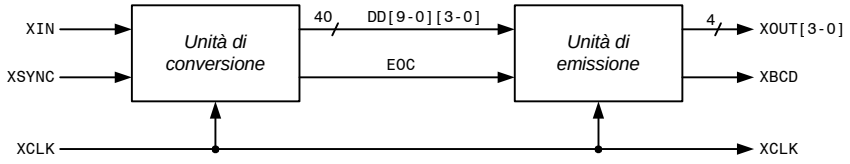


Fig. 8 L'interfaccia può essere partizionata in due unità: una *Unità di Conversione*, che provvede a convertire in base 10 il numero binario a 32 bit ricavato dall'ingresso `XIN`, e una *Unità di Emissione*, dedicata ad emettere serialmente verso l'esterno le cifre generate dalla conversione. Poiché il massimo numero decimale rappresentabile su 32 bit è 4294967295, che ha 10 cifre, i dati `DD` trasmessi dall'Unità di Conversione all'Unità di Emissione sono costituiti da 40 bit (10 cifre BCD da 4 bit ciascuna); un segnale di fine conversione (*End of Conversion*, `EOC`) verrà anche prodotto dall'Unità di Conversione al termine delle proprie operazioni per avviare l'attività dell'Unità di Emissione, che utilizzerà `EOC` come strobe per catturare i dati `DD`.

$$N = 2^{31}x_{31} + 2^{30}x_{30} + ... + 2^{2}x_{2} + 2^{1}x_{1} + 2^{0}x_{0} \quad \text{(a)}$$

$$N = x_0 + 2(x_1 + 2(x_2 + ... + 2(x_{30} + 2(x_{31}))...)) \quad \text{(b)}$$

Fig. 9 Il numero N rappresentato dai bit { x_{31} x_{30} ... x_1 x_0 } si determina valutando il polinomio in (*a*). L'espressione in (*b*), che usa la ben nota *regola di Horner* per la valutazione di un polinomio, fornisce lo stesso risultato, ma è particolarmente comoda in questa applicazione poiché l'ordine di utilizzo dei bit x_i è identico all'ordine con cui essi arrivano dall'ingresso `XIN`; il polinomio può di conseguenza essere valutato *on-line*, senza necessità di memorizzare preventivamente i bit del blocco. Quale che sia l'espressione utilizzata nell'implementazione della conversione, **è essenziale che le operazioni relative vengano eseguite in base 10**, ossia nella base in cui si vuole rappresentato il numero N.

0000000000 x 2 + **1** =
0000000001 x 2 + **0** =
0000000002 x 2 + **1** =
0000000005 x 2 + **0** =
0000000010 x 2 + **0** =
0000000020 x 2 + **1** =
0000000041 x 2 + **1** =
0000000083 x 2 + **1** =
0000000167 x 2 + **0** =
0000000334 x 2 + **0** =
0000000668 x 2 + **1** =
0000001337 x 2 + **0** =
0000002674 x 2 + **0** =
0000005348 x 2 + **1** =
0000010697 x 2 + **0** =
0000021394 x 2 + **1** =
0000042789 x 2 + **1** =
0000085579 x 2 + **0** =
0000171158 x 2 + **0** =
0000342316 x 2 + **1** =
0000684633 x 2 + **0** =
0001369266 x 2 + **0** =
0002738532 x 2 + **0** =
0005477064 x 2 + **1** =
0010954129 x 2 + **0** =
0021908258 x 2 + **0** =
0043816516 x 2 + **1** =
0087633033 x 2 + **1** =
0175266067 x 2 + **0** =
0350532134 x 2 + **0** =
0701064268 x 2 + **1** =
1402128537 x 2 + **1** =
2804257075

Fig. 10 Esempio di conversione binario-decimale sulla sequenza binaria { 1010 0111 0010 0101 1001 0001 0011 0011 }, corrispondente al numero decimale 2804257075. Si noti come ad ogni passo l'accumulatore contenga il risultato della conversione relativo a tutti i bit della sequenza applicati fino a quel momento.

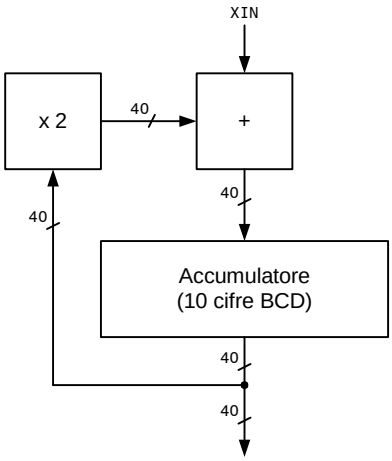


Fig. 11 L'espressione in Fig. 9*b* può essere implementata con un accumulatore che contiene le 10 cifre BCD, un moltiplicatore per 2 e un addizionatore. (Si sottolinea ancora una volta che sia il moltiplicatore che l'addizionatore devono operare in base 10, non in base 2.) Si osservi anche come l'algoritmo usato sia identico a quello applicato quando la conversione da base 2 a base 10 viene realizzata manualmente con carta e matita. L'accumulatore è inizializzato a 0, e ad ogni bit `XIN` entrante il suo contenuto viene aggiornato in modo da contenere la rappresentazione decimale della sequenza di bit applicata.

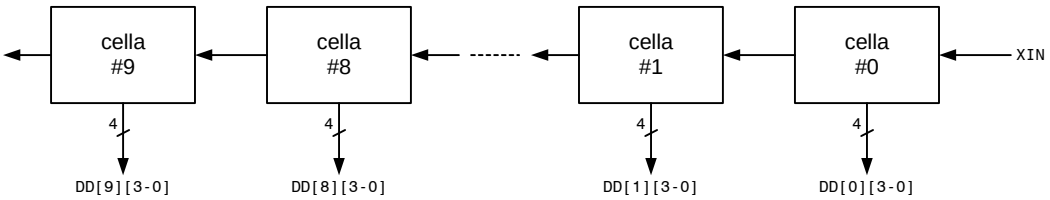


Fig. 12 In analogia con l'esecuzione manuale dell'algoritmo di conversione, dove ciascuna cifra viene trattata separatamente a partire dalla meno significativa, la struttura di Fig. 11 può essere realizzata come rete iterativa a 10 celle, dove ciascuna cella è dedicata alle operazioni su una singola cifra BCD. Come nelle normali operazioni manuali, il "riporto" tra una cella e la successiva può soltanto essere 0 o 1, e dunque può essere espresso su un solo bit; il bit in ingresso `XIN` può essere applicato come riporto di ingresso alla cella meno significativa; poiché il risultato non può mai eccedere le 10 cifre, il riporto finale dalla cella #9 rimane inutilizzato.

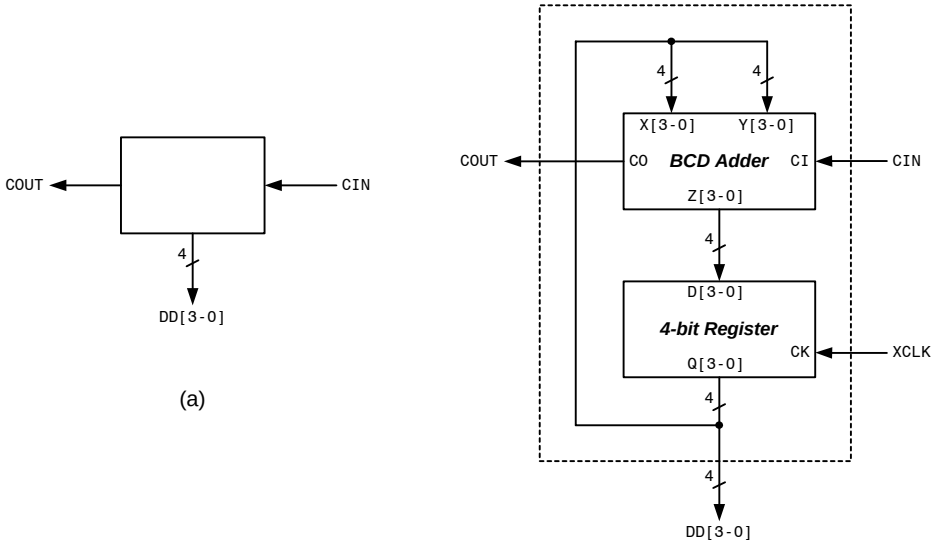


Fig. 13 La struttura della singola cella (*a*) è illustrata in (*b*). Un registro a 4 bit fa da accumulatore per una singola cifra decimale, mentre la moltiplicazione per 2 viene realizzata semplicemente applicando la cifra ad entrambi gli operandi dell'addizionatore BCD. I meccanismi per l'inizializzazione del registro BCD non sono indicati, ma saranno ripresi in considerazione ad implementazione ultimata.

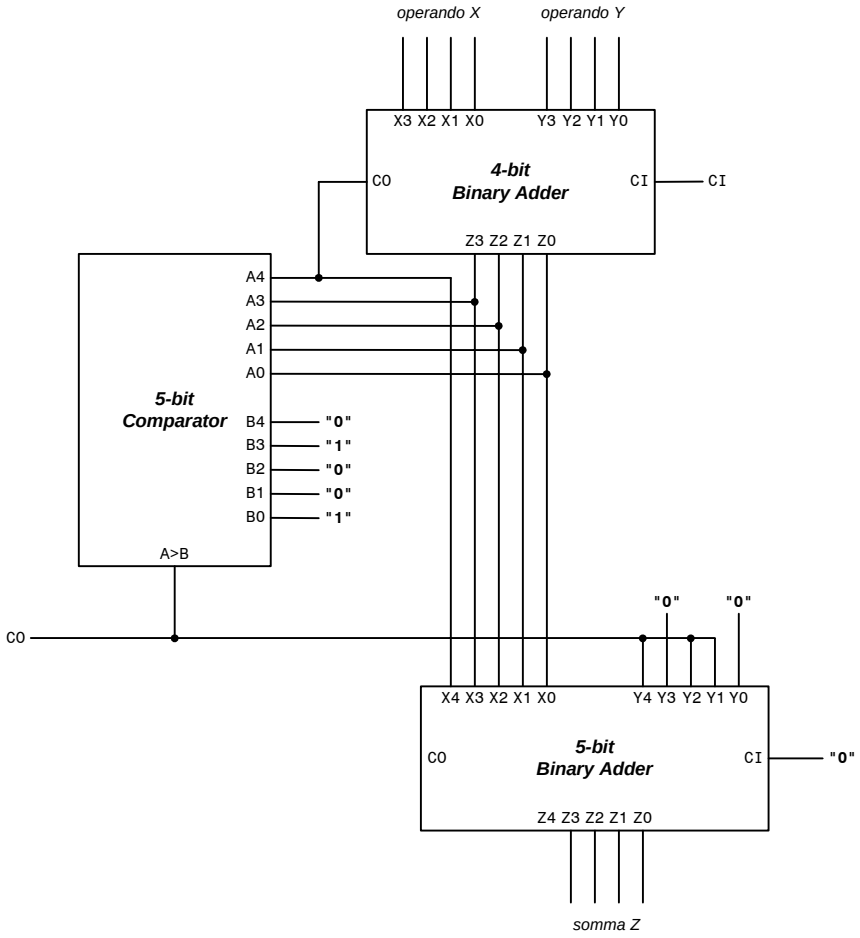


Fig. 14 Un addizionatore BCD (decimale) può in linea di principio essere realizzato col circuito qui illustrato. I due operandi BCD vengono applicati a un addizionatore binario a 4 bit, e il risultato, comprensivo di riporto finale, viene confrontato con la costante `01001` (decimale 9): se esso è maggiore di 9, gli viene sottratto 10 e viene generato un riporto di uscita `CO`, altrimenti il risultato originale passa immutato sull'uscita e `CO` = 0. La sottrazione condizionata viene realizzata mediante un secondo addizionatore binario a 5 bit, il cui operando Y viene posto a `00000` se il risultato non ha bisogno di correzione, mentre in caso contrario viene applicata la costante `10110` (complemento a 2 di $\texttt{01010}_2 = \texttt{10}_{10}$).

cifra corrente	DD[3-0]	CIN	*cifra successiva*	DD'[3-0]	COUT
0	0000	0	0	0000	0
0	0000	1	1	0001	0
1	0001	0	2	0010	0
1	0001	1	3	0011	0
2	0010	0	4	0100	0
2	0010	1	5	0101	0
3	0011	0	6	0110	0
3	0011	1	7	0111	0
4	0100	0	8	1000	0
4	0100	1	9	1001	0
5	0101	0	0	0000	1
5	0101	1	1	0001	1
6	0110	0	2	0010	1
6	0110	1	3	0011	1
7	0111	0	4	0100	1
7	0111	1	5	0101	1
8	1000	0	6	0110	1
8	1000	1	7	0111	1
9	1001	0	8	1000	1
9	1001	1	9	1001	1

Fig. 15 Il circuito di Fig. 14 è di principio, e potrebbe essere significativamente semplificato. Tuttavia la cella di conversione può essere realizzata senza far ricorso ad addizionatori, semplicemente osservando che essa si comporta in realtà come una macchina sequenziale (di Mealy) con un ingresso `CIN` e un'uscita `COUT`, il cui stato codifica una cifra BCD e il cui comportamento è illustrato nella tavola di transizione qui sopra. Ad esempio, se la cella contiene la cifra 6 ed è presente riporto `CIN` in ingresso, allora, poiché $2\times 6+1=13$, essa deve generare un riporto `COUT=1` in uscita e predisporsi affinché il suo stato successivo sia quello corrispondente alla cifra 3.

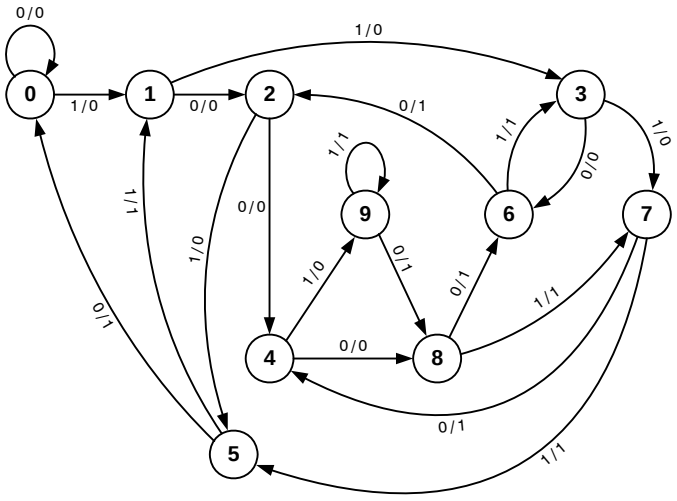


Fig. 16 Diagramma di stato per la macchina sequenziale di Fig. 15.

DD[3-0]	DD'[3-1]	COUT
0000	000	0
0001	001	0
0010	010	0
0011	011	0
0100	100	0
0101	000	1
0110	001	1
0111	010	1
1000	011	1
1001	100	1
1010	---	-
1011	---	-
1100	---	-
1101	---	-
1110	---	-
1111	---	-

Fig. 17 Tavola di transizione per la macchina di Fig. 15, nell'ipotesi di realizzazione con flip-flop D. La tavola è ridotta rispetto a quella riportata in Fig. 15, osservando che `D'[0]` dipende solo da `CIN` (essendo `D'[0] = CIN`) e che `CIN` non ha effetti né su `D'[3-1]` né su `COUT`. Le ultime 6 righe corrispondono a configurazioni illegali nel codice BCD, e danno luogo a condizioni non specificate sia per lo stato futuro che per l'uscita.

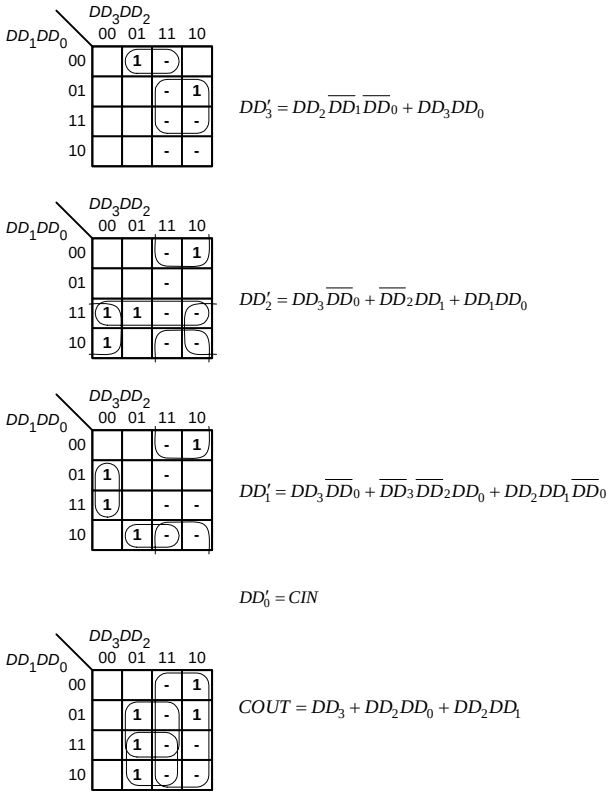


Fig. 18 Mappe di Karnaugh per la tavola di Fig. 17.

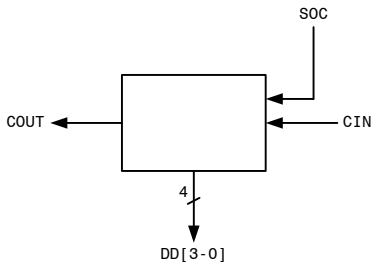


Fig. 19 La cella deve ricevere anche un segnale di *inizio conversione* (Start of Conversion, `SOC`), che dovrà essere applicato all'arrivo di `XSYNC`, ossia alla presentazione del primo dei 32 bit da convertire. `XSYNC` può apparire immediatamente dopo la fine del blocco precedente (Fig. 4*a*), dunque tale inizializzazione deve essere di tipo sincrono, poiché i bit `XIN` devono poter entrare nella rete iterativa senza soluzione di continuità. L'inizializzazione della cella comporta che l'accumulatore venga avviato dallo stato $\texttt{000...000}_{10}$ oppure $\texttt{000...001}_{10}$ a seconda che sia `XIN` = 0 oppure `XIN` = 1; tenendo presente la struttura di Fig. 12, ciò si realizza facilmente imponendo che ciascuna cella venga inizializzata in modo che (1) lo stato futuro sia `0000` oppure `0001` a seconda che sia `CIN` = 0 oppure `CIN` = 1, (2) debba essere `COUT` = 0.

$$
\begin{aligned}
DD'_3 &= \overline{SOC}(DD_2\overline{DD}_1\overline{DD}_0 + DD_3DD_0)\\
DD'_2 &= \overline{SOC}(DD_3\overline{DD}_0 + \overline{DD}_2DD_1 + DD_1DD_0)\\
DD'_1 &= \overline{SOC}(DD_3\overline{DD}_0 + \overline{DD}_3\overline{DD}_2DD_0 + DD_2DD_1\overline{DD}_0)\\
DD'_0 &= CIN\\
COUT &= \overline{SOC}(DD_3 + DD_2DD_0 + DD_2DD_1)
\end{aligned}
$$

Fig. 20 Le equazioni di Fig. 18 si modificano di conseguenza.

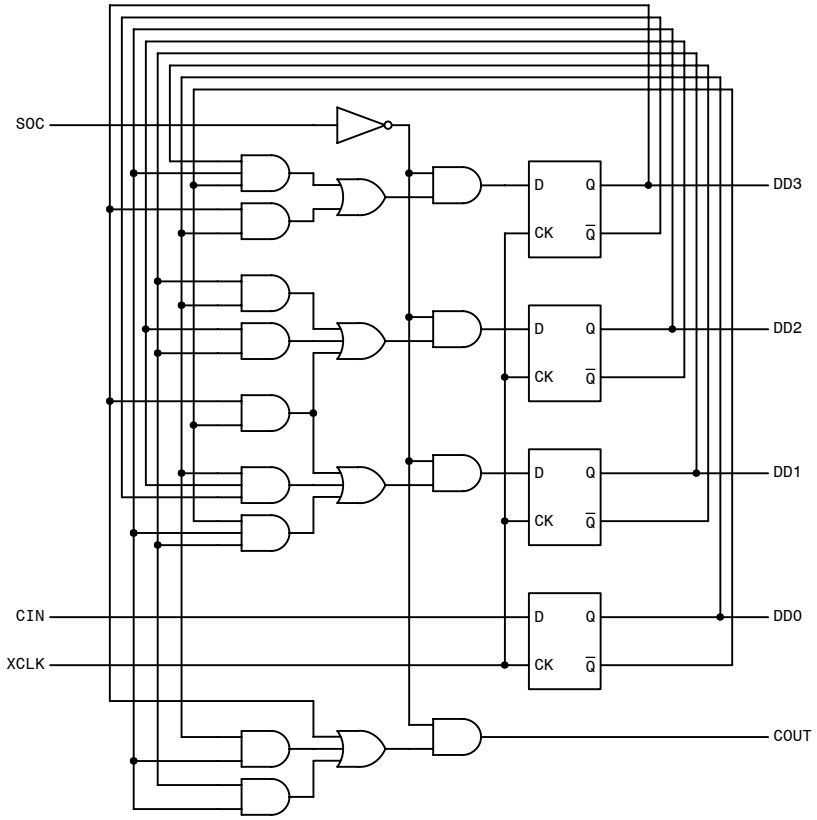


Fig. 21 Realizzazione della cella di conversione.

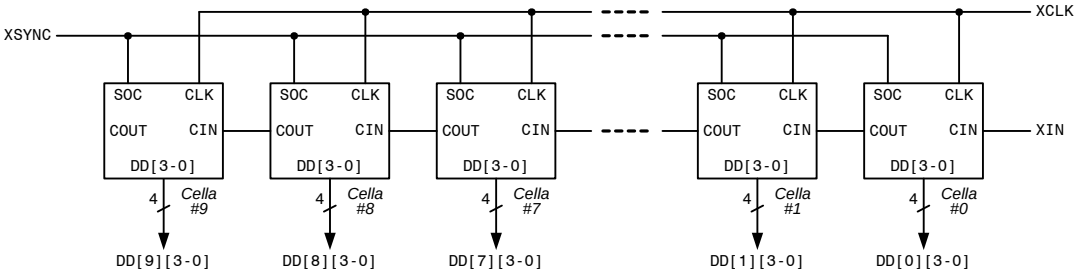
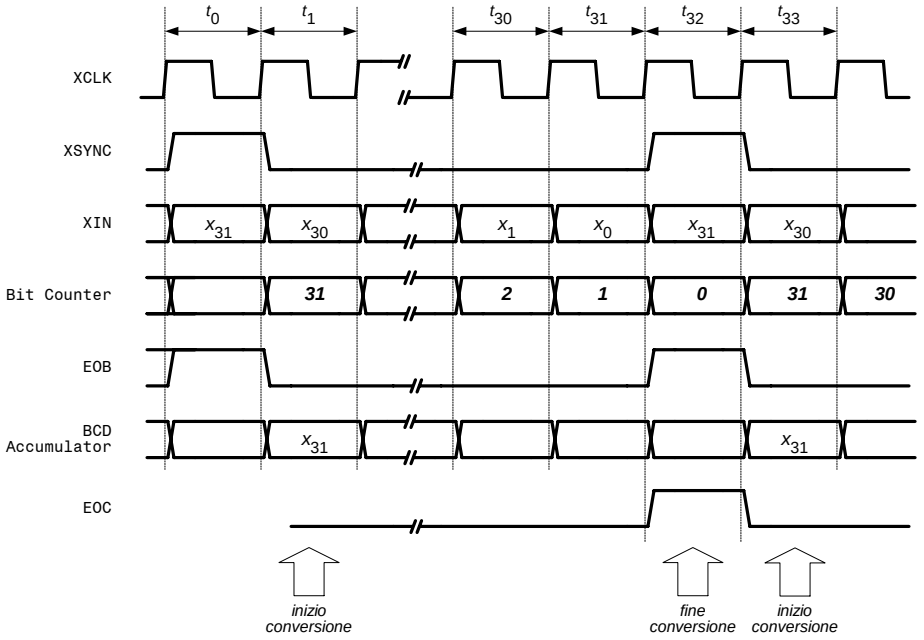
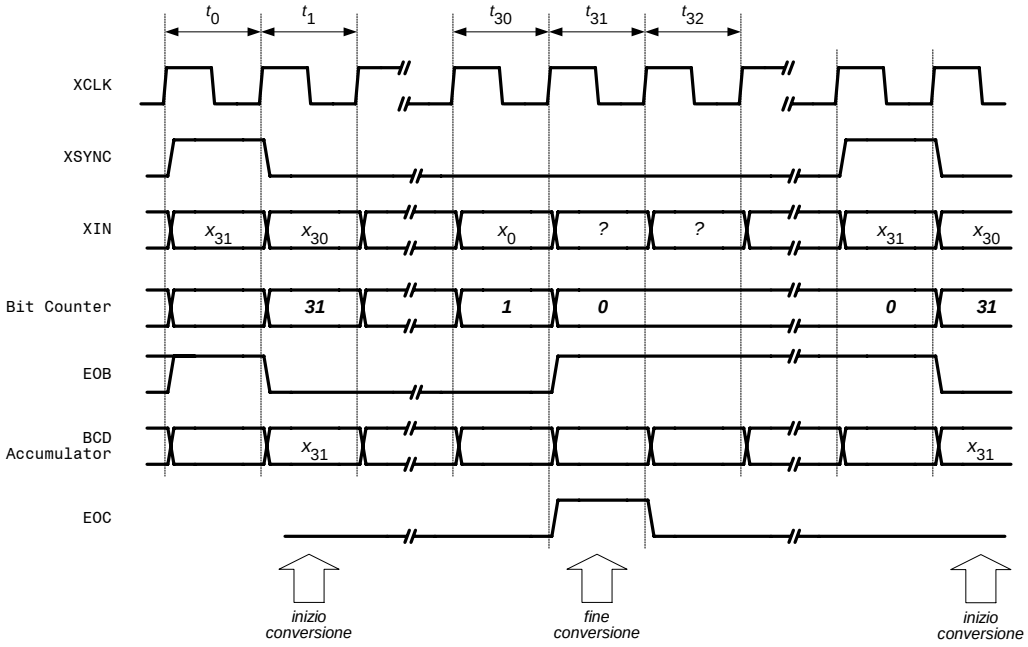


Fig. 22 Realizzazione finale dell'Unità di Conversione binario-decimale.

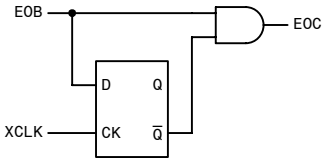


(a)

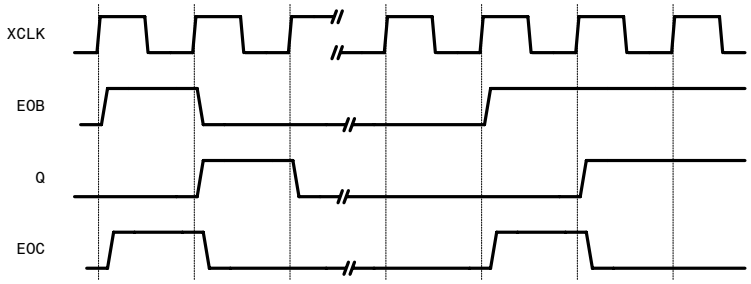


(b)

Fig. 23 Temporizzazioni del convertitore binario-decimale. In (*a*) è rappresentato il caso di blocchi contigui: nell'intervallo t_1 il bit x_{31} è entrato nell'accumulatore BCD, nell'intervallo t_{32} l'accumulatore contiene il risultato della conversione; il segnale EOC (End of Conversion) che andrà generato a beneficio dell'Unità di Emissione, è solo apparentemete coincidente con EOB. In realtà, come illustrato in (*b*) nel caso di blocchi non contigui, EOC deve avere sempre durata di un singolo periodo di XCLK; infatti, non disponendo le celle di conversione di un segnale di Enable, il contenuto dell'accumulatore cambia ad ogni XCLK, e di conseguenza esso contiene risultati validi per un solo periodo di XCLK.



(a)



(b)

Fig. 24 Generazione del segnale EOC (End of Conversion), secondo i requisiti illustrati in Fig. 23.

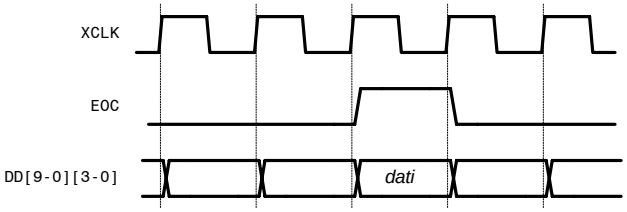


Fig. 25 A questo punto, l'Unità di Emissione riceve i 40 bit (corrispondenti a 10 cifre in codice BCD a 4 bit) dell'accumulatore BCD, unitamente al segnale EOC che qualifica la validità dei dati. L'Unità dovrà immagazzinare i dati BCD ricevuti, in modo da consentire all'Unità di Conversione l'eventuale avvio della conversione su un blocco successivo.

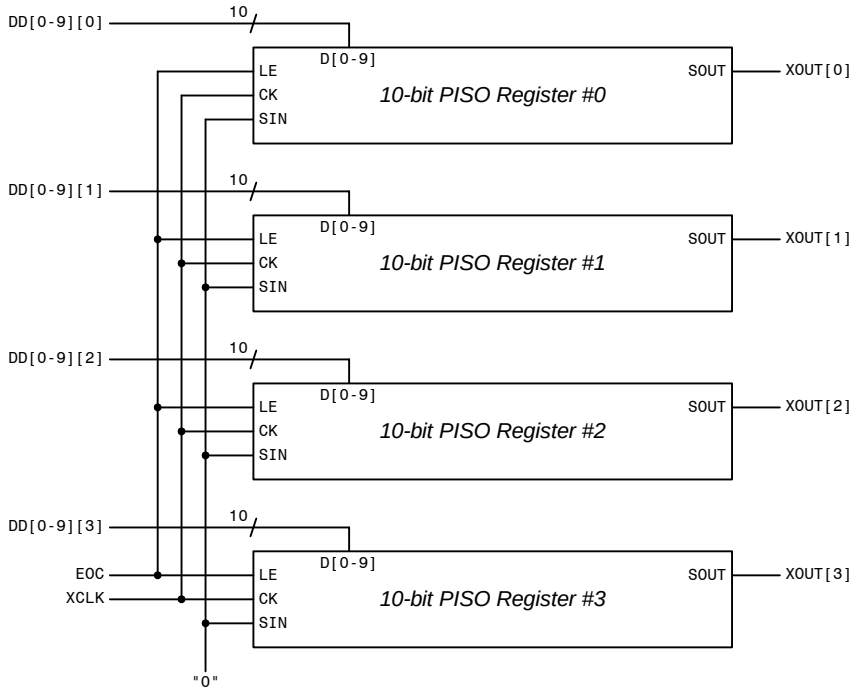


Fig. 26 Una possibile architettura dell'Unità di Emissione è basata su quattro registri Parallel-In Serial-Out (PISO), uno per ciascun bit della cifra BCD da emettere serialmente. I registri vengono caricati in parallelo mediante il segnale `EOC` di fine conversione, ed emettono serialmente il loro contenuto verso le uscite `XOUT`. Si notino le connessioni tra le linee `DD[9-0][3-0]` e gli ingressi dati `D[9-0]` dei registri PISO, importanti affinché le cifre BCD vengano emesse nel corretto ordine: assumendo che l'ingresso dati `D0` corrisponda alla cella più a sinistra del PISO e l'ingresso dati `D9` corrisponda alla cella più a destra, il registro #0 riceverà la linea `DD[0][0]` sull'ingresso `D0` e via via ordinatamente fino a ricevere la linea `DD[9][0]` sull'ingresso `D9`. In generale, l'ingresso `Dj` del PISO #*k* dovrà ricevere la linea `DD[j][k]`.

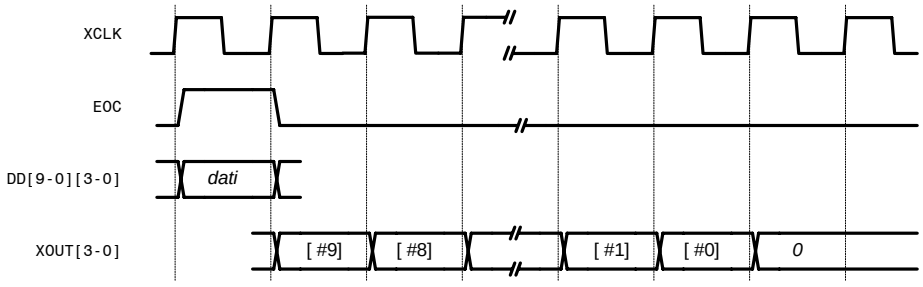


Fig. 27 Temporizzazioni della struttura di Fig. 26. Essendo sincrono il caricamento dei registri PISO, la cifra più significativa (#9) del numero BCD appare nel periodo di clock successivo a quello in cui viene applicato `EOC`.

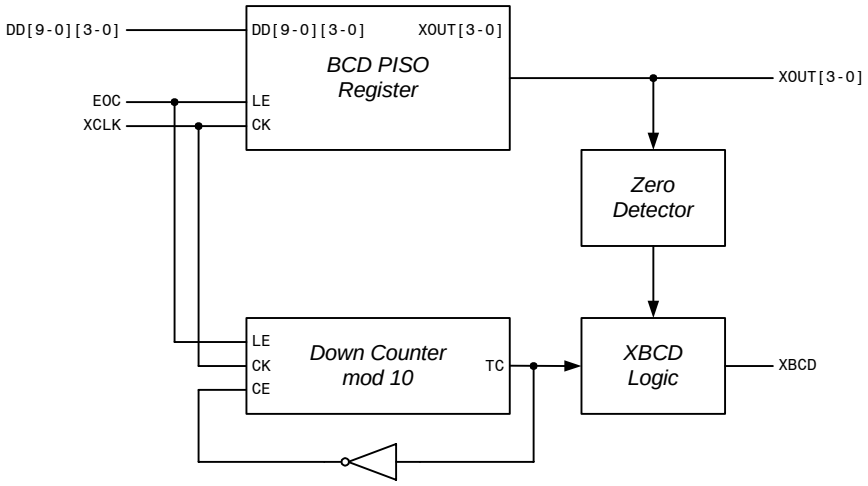


Fig. 28 La generazione del segnale di uscita XBCD può essere risolta nella struttura di Fig. 26 (il cui circuito è qui rappresentato dal modulo BCD PISO Register) con un circuito in grado di rivelare se la cifra BCD in corso di emissione è uno zero. Dal momento che, se le cifre emesse sono tutte costituite da zeri, XBCD dovrà essere attivo sull'ultima di esse, sarà necessario adottare un contatore modulo 10 il cui Terminal Count forzi in ogni caso XBCD attivo; per evitare Terminal Count multipli, il contatore viene bloccato sul conteggio finale; inoltre, affinché esso proceda in sincronismo con le cifre emesse, la sua inizializzazione è di tipo sincrono ed è asservita al segnale EOC di fine conversione. Una opportuna logica provvederà poi a generare il segnale XBCD. (Questa soluzione, in realtà, aggira in qualche modo il requisito imposto dal testo del problema, nel senso che gli zeri non significativi vengono comunque emessi, ma in corrispondenza ad essi XBCD viene lasciato inattivo.)

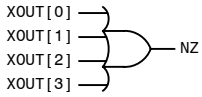


Fig. 29 Il rivelatore di zero è realizzato mediante un OR a 4 ingressi: l'uscita sarà 1 quando la cifra BCD presente agli ingressi è diversa da zero.

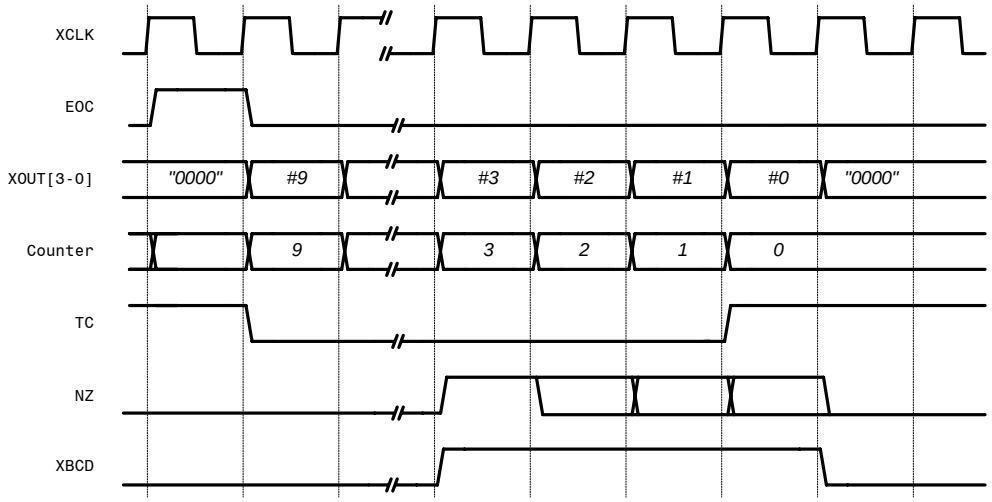
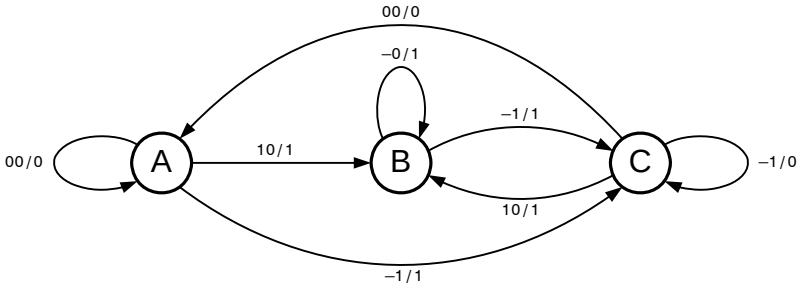


Fig. 30 La temporizzazione illustra il comportamento desiderato per XBCD (nell'esempio, tutte le cifre dalla #9 alla #4 sono uguali a zero): esso deve diventare attivo non appena NZ=1, e deve tornare inattivo al termine dell'emissione dell'ultima cifra (#0). Si noti come l'aver lasciato che i BCD PISO Register si riempiano con zeri tramite i rispettivi Serial Input, garantisca che NZ diventi inattivo quando i registri si sono svuotati.

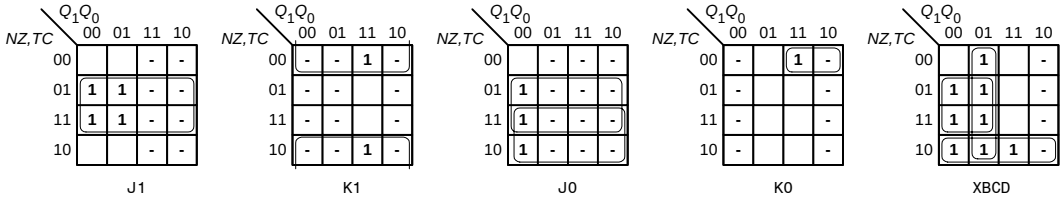


(a)

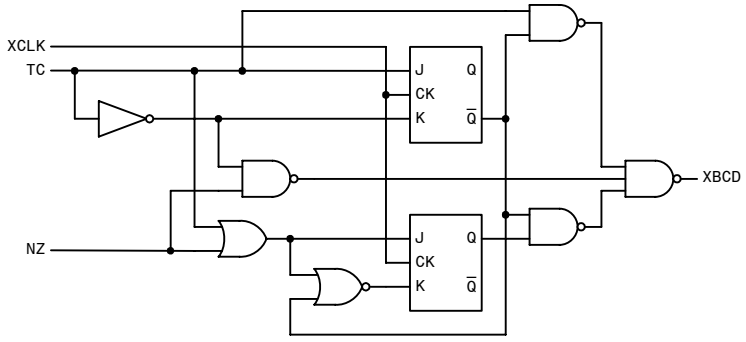
Q1	Q0	NZ	TC	Q1'	Q0'	XBCD	J1	K1	J0	K0
A	0	0	0 0	A	0	0	0	0 -	0 -	
A	0	0	0 1	C	1	1	1	1 -	1 -	
A	0	0	1 0	B	0	1	1	0 -	1 -	
A	0	0	1 1	C	1	1	1	1 -	1 -	
B	0	1	0 0	B	0	1	1	0 -	- 0	
B	0	1	0 1	C	1	1	1	1 -	- 0	
B	0	1	1 0	B	0	1	1	0 -	- 0	
B	0	1	1 1	C	1	1	1	1 -	- 0	
-	1	0	- -	-	-	-	-	- -	- -	
C	1	1	0 0	A	0	0	0	- 1	- 1	
C	1	1	0 1	C	1	1	0	- 0	- 0	
C	1	1	1 0	B	0	1	1	- 1	- 0	
C	1	1	1 1	C	1	1	0	- 0	- 0	

(b)

Fig. 31 Il circuito generatore di XBCD è una macchina sequenziale, il cui diagramma di stato è illustrato in (*a*): gli ingressi sono NZ e TC, mentre l'uscita è XBCD. Nello stato **A** di riposo si aspetta l'arrivo di un NZ (dopo il quale si passa allo stato **B**) oppure di TC (dopo il quale si passa nello stato **C**); il ritorno a riposo si ha quando TC torna a 0 all'inizio di una nuova emissione. Seguendo attentamente le transizioni, si vedrà come tutti i casi particolari siano contemplati, compresi quello in cui tutte le cifre sono zero e quello in cui NZ si presenta contemporaneamente a TC. In (*b*) sono rappresentate la tavola di transizione e la tavola di eccitazione nel caso di implementazione con flip-flop JK.



(a)



(b)

Fig. 32 In (*a*) sono rappresentate le mappe di Karnaugh per gli ingressi J, K e l'uscita XBCD, mentre in (*b*) appare una possibile implementazione del generatore di XBCD.

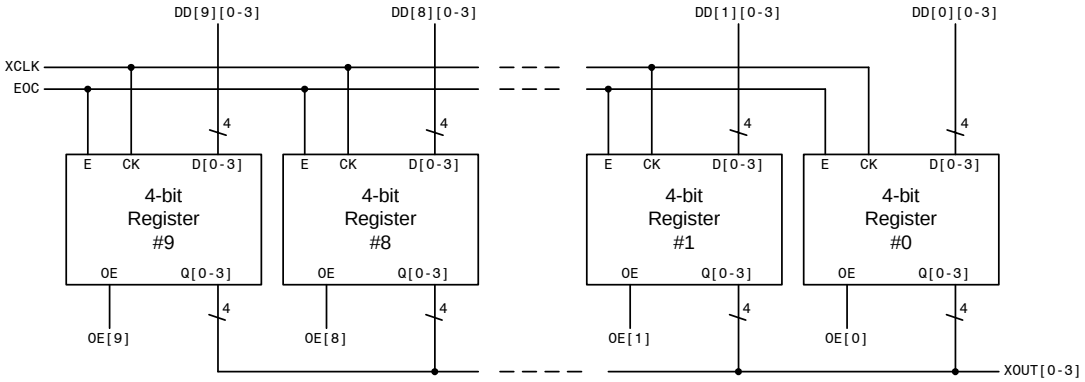


Fig. 33 Un'architettura alternativa per l'Unità di Emissione è invece basata su 10 registri D a 4 bit ciascuno, con uscita tri-state. Ciascun registro riceve una cifra BCD, e l'ordine di attivazione delle uscite determina l'ordine di emissione delle cifre sulle uscite XOUT. Una tale struttura richiederà un contatore e un decoder per l'attivazione delle uscite. Si osservi come, mentre nella struttura di Fig. 26 le cifre BCD vengono emesse sempre a partire dalla #9, qui è possibile iniziare l'emissione a partire da una cifra qualunque.

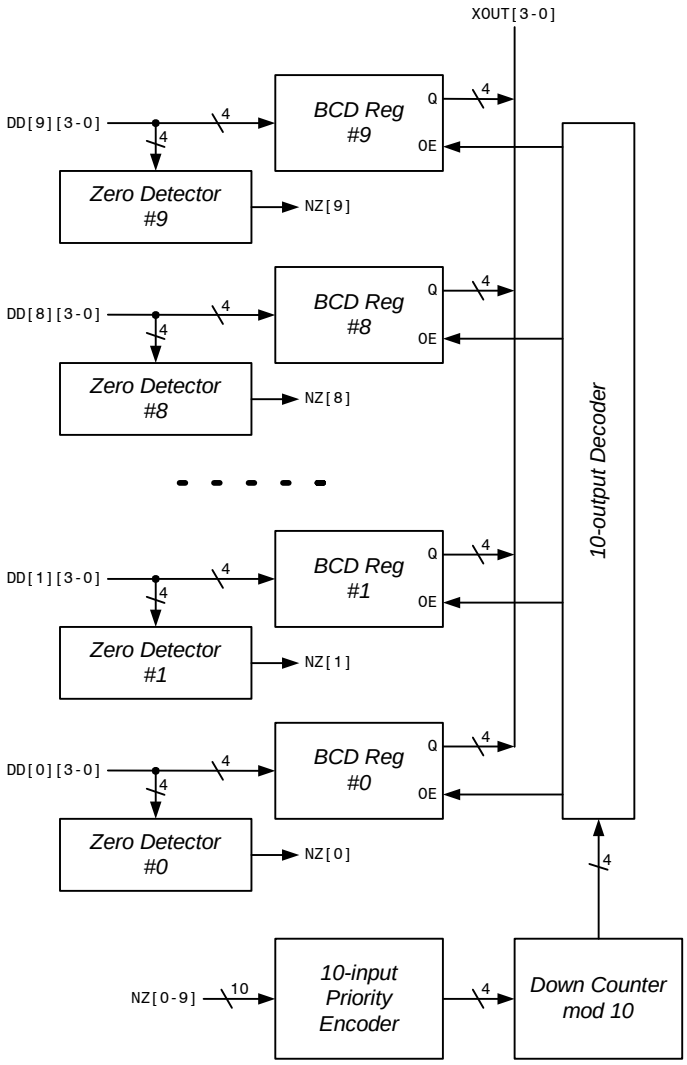


Fig. 34 La serializzazione delle cifre e la generazione di `XBCD` per la seconda architettura dell'Unità di Emissione (Fig. 33) si basa sull'idea seguente: si usi un rivelatore di zero per ciascuna cifra BCD generata dall'accumulatore (e che quindi potrebbe essere addirittura incorporato nella cella di conversione), si inviino le 10 uscite `NZ[0-9]` a un codificatore a priorità, il quale genererà l'indice della prima cifra diversa da zero, si carichi con questo indice un Down Counter modulo 10, il quale piloti un decoder le cui uscite vadano infine ad attivare ordinatamente le uscite tri-state dei BCD Registers. Se ad esempio la prima cifra diversa da zero è la #7, il priority encoder genererà il codice 0111, il counter inizierà a contare da 7 e man mano che il suo conteggio decrementa, verranno attivate le uscite tri-state dei BCD Register #7, poi #6, e così via fino al #0. Per garantire che almeno una cifra, l'ultima, venga sempre emessa, è sufficiente non usare l'uscita `NZ[0]` e applicare un 1 fisso al corrispondente ingresso del priority encoder. Il caricamento del Down Counter può avvenire nel momento stesso in cui vengono caricati i BCD Registers, ossia in corrispondenza al segnale `EOC` di fine conversione. Contrariamente alla soluzione precedente, in questo caso gli zeri non significativi *non vengono mai emessi*.

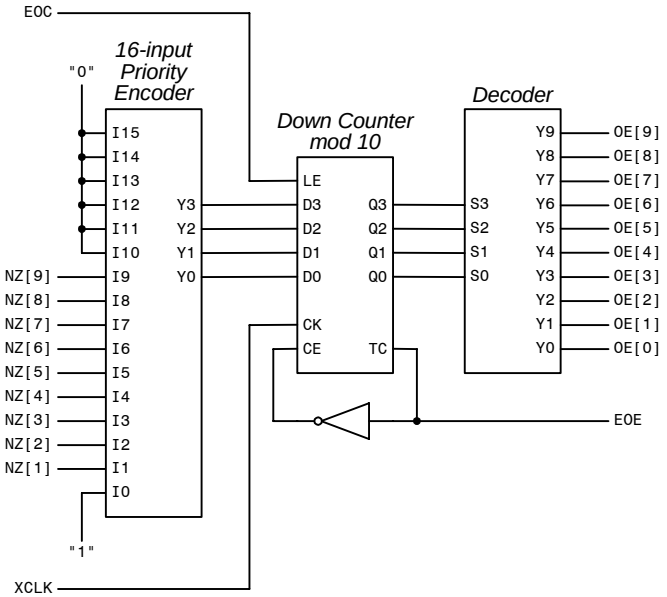


Fig. 35 Il relativo circuito è qui illustrato. I rivelatori di zero, che sono tutti identici al circuito di Fig. 29, non sono rappresentati. Il Terminal Count del Down Counter fornisce un segnale di fine emissione (End of Emission, EOE), che verrà più avanti utilizzato per la generazione di XBCD.

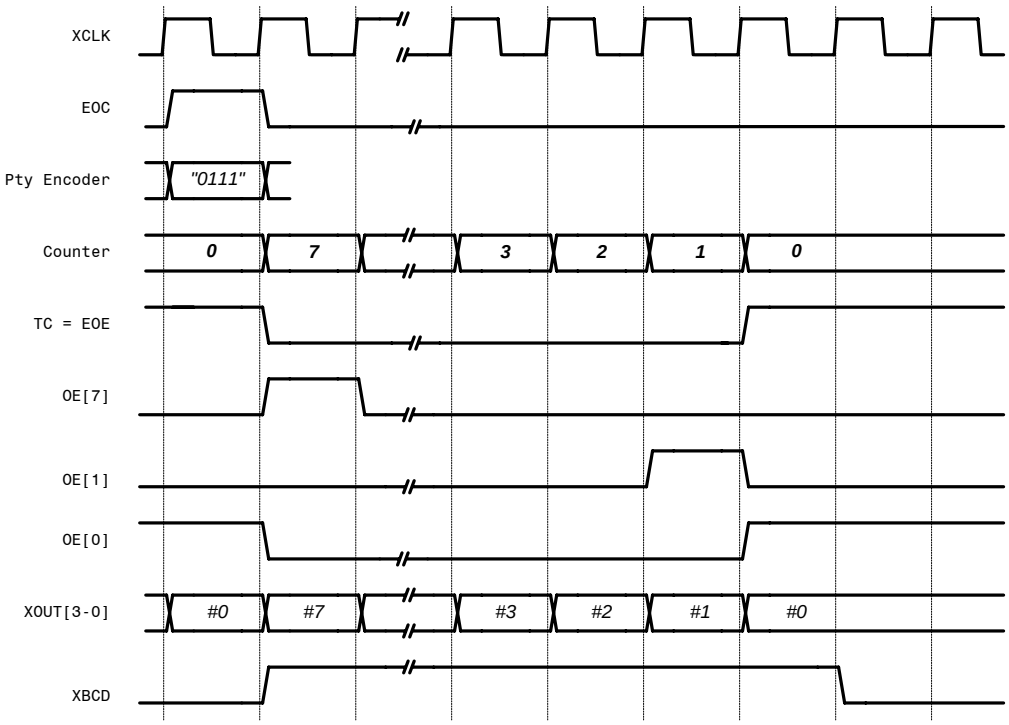
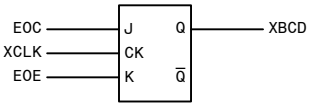
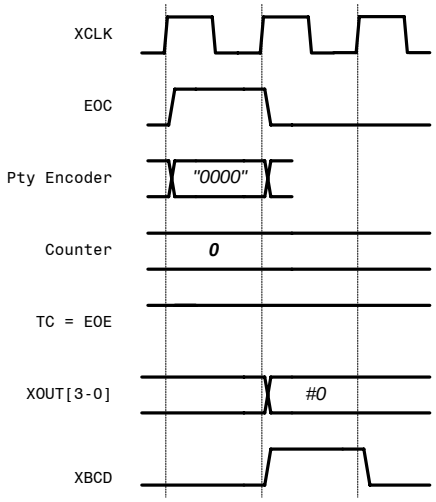


Fig. 36 Temporizzazioni della struttura di Fig. 35, nell'ipotesi che la cifra #7 sia la prima diversa da zero. La generazione di XBCD è adesso particolarmente semplice: tale segnale deve diventare attivo nel periodo di clock successivo ad EOC, e tornare inattivo al primo clock dopo EOE.



(a)



(b)

Fig. 37 In (*a*) è raffigurata la generazione del segnale XBCD mediante un semplice flip-flop JK. In (*b*) è illustrato il comportamento del circuito nel caso limite in cui solo l'ultima cifra BCD debba essere emessa.