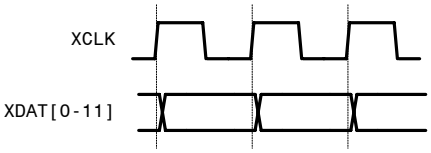
Interfaccia IFHIST



Un'interfaccia IFHIST, connessa al bus di I/O PD-32, riceve in continuazione da un canale esterno `XDAT[0-11]` dati numerici a 12 bit, sincronizzati a un clock `XCLK` a 10 MHz (si veda la figura a fianco). Alla ricezione da parte della CPU di un numero N (con $0 < N \leq 8192$) l'interfaccia, dopo aver eseguito le opportune inizializzazioni, acquisisce un blocco di N dati consecutivi dal canale esterno e ne calcola l'*istogramma*, definito come un vettore **H** la cui componente k-esima indica quanti dati del blocco hanno valore k. Al termine delle operazioni, l'interfaccia invia un'interruzione alla CPU e rende ad essa disponibile l'istogramma calcolato.

Progettare l'hardware dell'interfaccia, e illustrare le relative temporizzazioni.

(*) Progetto d'esame per il corso di Reti Logiche, appello del 2007-03-19, Laurea Specialistica in Ingegneria Informatica, Università di Roma "La Sapienza".

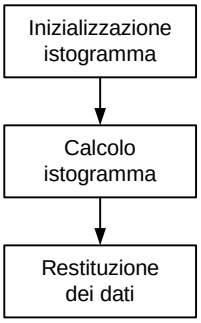


Fig. 1 Allo Start, l'interfaccia deve passare attraverso tre fasi distinte:
(1) fase di *inizializzazione*, in cui vengono azzerati gli elementi del vettore istogramma;
(2) fase di *calcolo dell'istogramma*, in cui vengono aggiornati gli elementi del vettore istogramma in base ai dati ricevuti sul canale XDAT;
(3) fase di *restituzione dei dati*, in cui gli elementi del vettore istogramma vengono messi a disposizione della CPU.

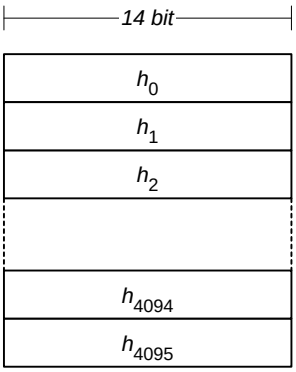


Fig. 2 Essendo XDAT un canale a 12 bit, i dati possono assumere valori da 0 a 4095; di conseguenza l'istogramma sarà un vettore a 4096 componenti $\mathbf{H} = \{h_0, h_1, \ldots, h_{4095}\}$, che dovranno essere immagazzinate in altrettanti accumulatori. Poiché la massima lunghezza della sequenza da acquisire è $N = 8192$, ciascun accumulatore dovrà poter contenere un qualsiasi numero da 0 (caso in cui *nessun* elemento della sequenza acquisita assume un dato valore) fino a 8192 compreso (caso in cui *tutti* gli elementi della sequenza acquisita assumono lo stesso valore); di conseguenza, ciascun accumulatore dovrà avere estensione di (almeno) 14 bit.

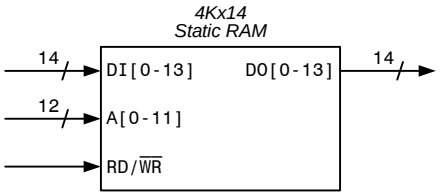


Fig. 3 Essendo del tutto irrealistico l'uso di 4096 registri indipendenti, gli accumulatori saranno contenuti all'interno di una RAM statica da 4096 parole da 14 bit ciascuna, che diventa così l'elemento centrale del Sottosistema di Calcolo (SCA).

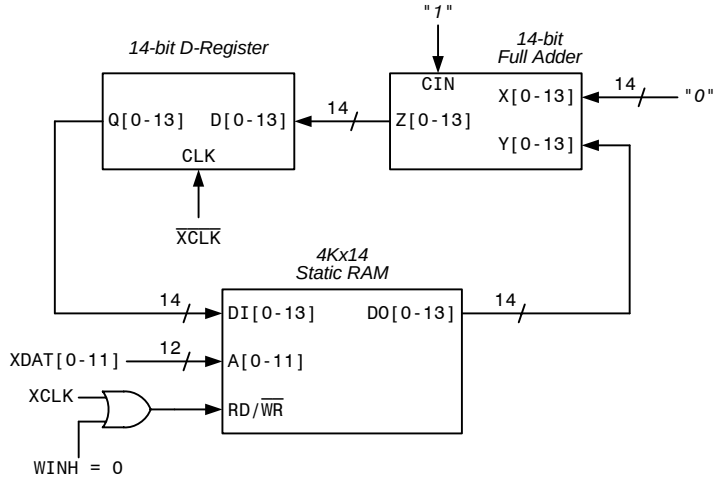


Fig. 4 Durante la fase di aggiornamento dell'istogramma, la RAM sarà configurata in modo da ricevere come indirizzo il dato XDAT in ingresso, il cui valore andrà a selezionare l'accumulatore corrispondente; il contenuto corrente dell'accumulatore viene letto e incrementato di 1 mediante un Full Adder a 14 bit; il risultato dell'incremento viene temporaneamente memorizzato in un registro e viene poi scritto nuovamente nella stessa locazione di RAM. Il registro è indispensabile per impedire l'instaurarsi di un loop combinatorio (gli elementi di memoria di una RAM statica sono essenzialmente dei latch). L'intera sequenza di operazioni deve avvenire in un singolo periodo di XCLK, pertanto nella prima metà del periodo (XCLK = 1) viene eseguita la lettura dell'accumulatore e l'incremento del suo valore, alla transizione $1 \rightarrow 0$ di XCLK il risultato viene catturato nel registro (da cui l'uso di $\overline{\text{XCLK}}$ come clock), e nella seconda metà (XCLK = 0) viene eseguita la riscrittura dell'accumulatore. La porta OR controllata da un segnale WINH (*Write Inhibit*) è indispensabile, dato che, nel corso delle operazioni dell'interfaccia, vi saranno evidentemente dei periodi di XCLK in cui *non* devono avvenire scritture in RAM; il segnale WINH, di conseguenza, sarà gestito dal Sottosistema di Controllo (SCO). (Il registro D potrebbe essere sostituito, come si vedrà più avanti, da un D-Latch a 14 bit, con ingresso di Gate controllato da XCLK anziché da $\overline{\text{XCLK}}$.)

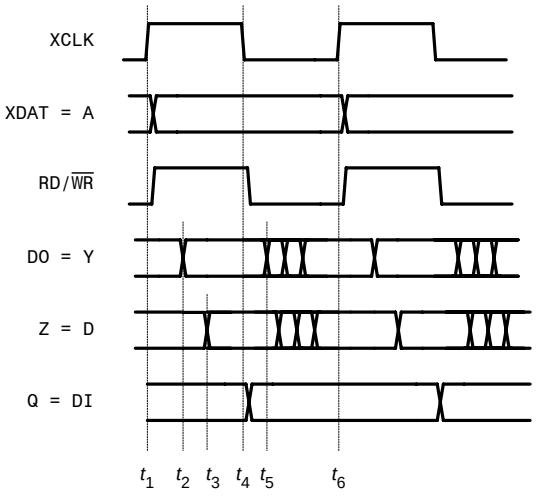


Fig. 5 Temporizzazione dei segnali nel circuito di Fig. 4. L'istante t_1 in cui appare il fronte di salita di XCLK, identifica l'inizio del ciclo; il segnale RD/$\overline{\text{WR}}$ segue l'andamento di XCLK con un ritardo dovuto alla porta OR. All'istante t_2 l'uscita dati dalla RAM si stabilizza dopo l'applicazione del nuovo indirizzo XDAT. All'istante t_3 viene completato l'incremento del dato letto dalla RAM. All'istante t_4 (fronte di discesa di XCLK) il valore incrementato viene catturato nel registro e viene contemporaneamente iniziata la scrittura in RAM. All'istante t_5 il dato in uscita dalla RAM riflette il valore attualmente applicato al suo ingresso dati (se non fosse presente il registro, tale variazione implicherebbe una nuova successiva variazione dell'uscita del Full Adder). All'istante t_6 (corrispondente a t_1 per il ciclo successivo) termina la fase di scrittura e contemporaneamente viene presentato un nuovo indirizzo alla RAM.

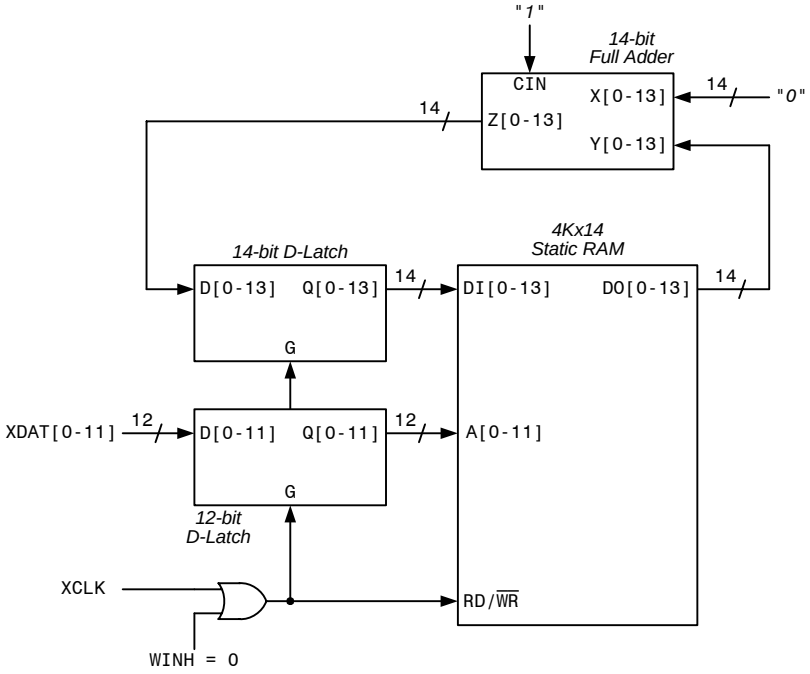


Fig. 6 Si noti, nel timing di Fig. 5, la criticità in corrispondenza a t_6: per via della porta OR su , XCLK torna a 1 prima che RD/$\overline{\text{WR}}$ torni a 1, di conseguenza esiste la possibilità che sia XDAT (indirizzo della RAM) sia l'uscita del Full Adder commutino prima che il ciclo di scrittura sia concluso – condizione che si verifica solo se i circuiti sono sufficientemente veloci. Tuttavia, per scongiurare *in ogni caso* tale possibilità, è possibile garantire la stabilità sia degli indirizzi che dei dati in ingresso alla RAM utilizzando dei latch sulle linee corrispondenti: quando la scrittura è attiva, i latch mantengono invariato il loro contenuto, che può mutare solo a scrittura avvenuta, quando cioè RD/$\overline{\text{WR}}$ torna a 1. Si noti come il latch degli indirizzi si comporti in modo trasparente quando non è richiesta la scrittura su RAM (WINH = 1).

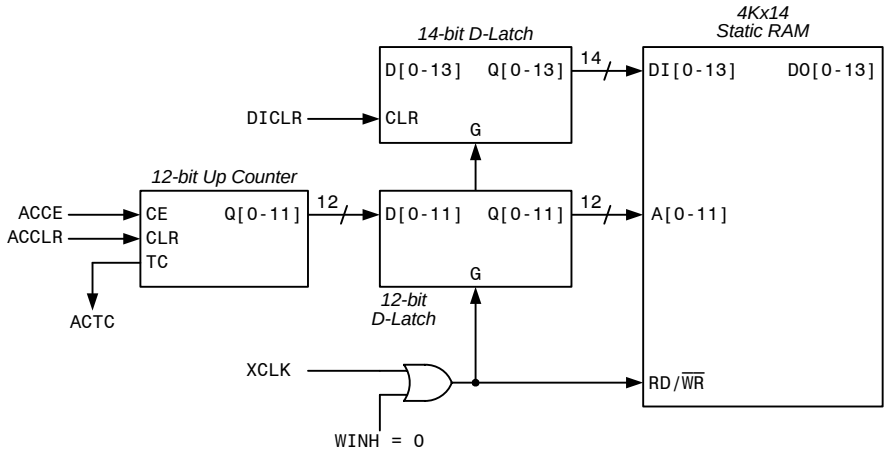


Fig. 7 Durante la fase di inizializzazione, tutte le locazioni della RAM devono essere poste a 0. La struttura di Fig. 6 può essere in gran parte riutilizzata allo scopo, forzando il Clear del latch dei dati in ingresso alla RAM e applicando ad essa, come indirizzi, le uscite di un contatore a 12 bit (Address Counter). Ad ogni ciclo di `XCLK`, in tal modo, viene scritto `000...0` entro ciascuna locazione di RAM. I segnali `DICLR` (Data Input Clear), `ACCE` (Address Counter Count Enable), `ACCLR` (Address Counter Clear) e `ACTC` (Address Counter Terminal Count) verranno gestiti dal SCO.

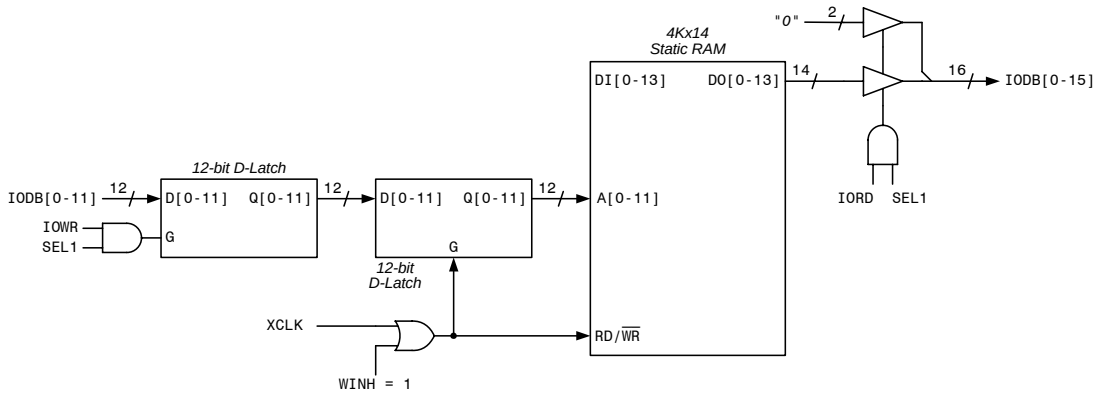


Fig. 8 Durante la fase di restituzione dei dati, la CPU deve poter selezionare uno dei 4096 accumulatori contenuti in RAM e leggerne il contenuto. Di conseguenza, la CPU invierà l'indirizzo dell'accumulatore su una porta di output a 12 bit, tale indirizzo verrà applicato alla RAM, e il dato in uscita da quest'ultima andrà applicato ad una porta di input a 14 bit (completati a 16 bit con due bit forzati a 0 nella parte più significativa). Si noti come le scritture in RAM siano inibite da `WINH` = 1. (Un interfacciamento più complesso avrebbe potuto essere realizzato utilizzando la RAM come blocco di *memoria condivisa* con la memoria principale del PD-32 attraverso il Memory Bus.)

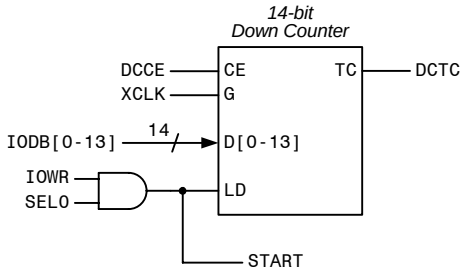


Fig. 9 La lunghezza del blocco di dati su cui andrà calcolato l'istogramma è un numero a 14 bit (poiché il valore 8192 è *compreso* nel range di lunghezze), che viene trasmesso dalla CPU attraverso una porta di output. La porta è costituita da un 14-bit Down Counter che verrà utilizzato per contare i dati del blocco; il conteggio iniziale viene impostato mediante parallel load asincrono. Il segnale di START viene derivato dal comando di caricamento del counter; i segnali DCCE (Data Counter Count Enable) e DCTC (Data Counter Terminal Count) verranno gestiti dal SCO.

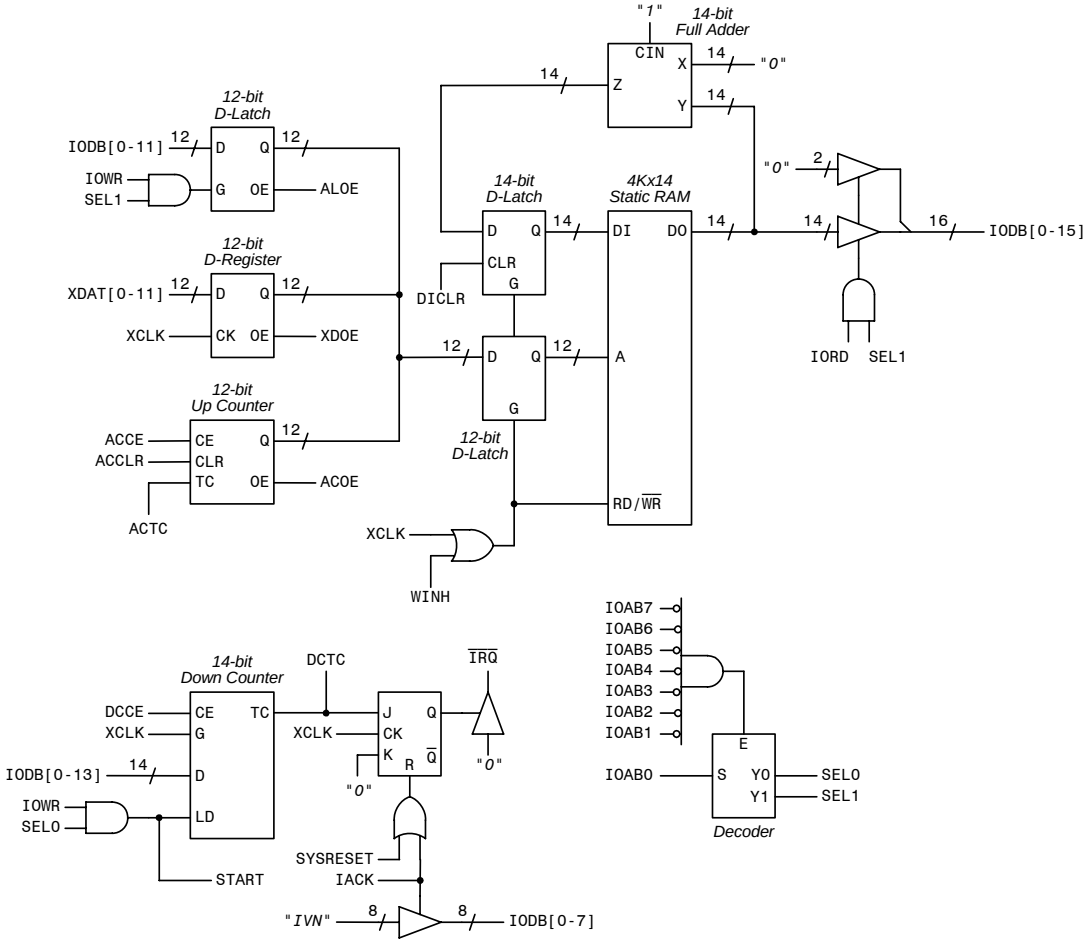
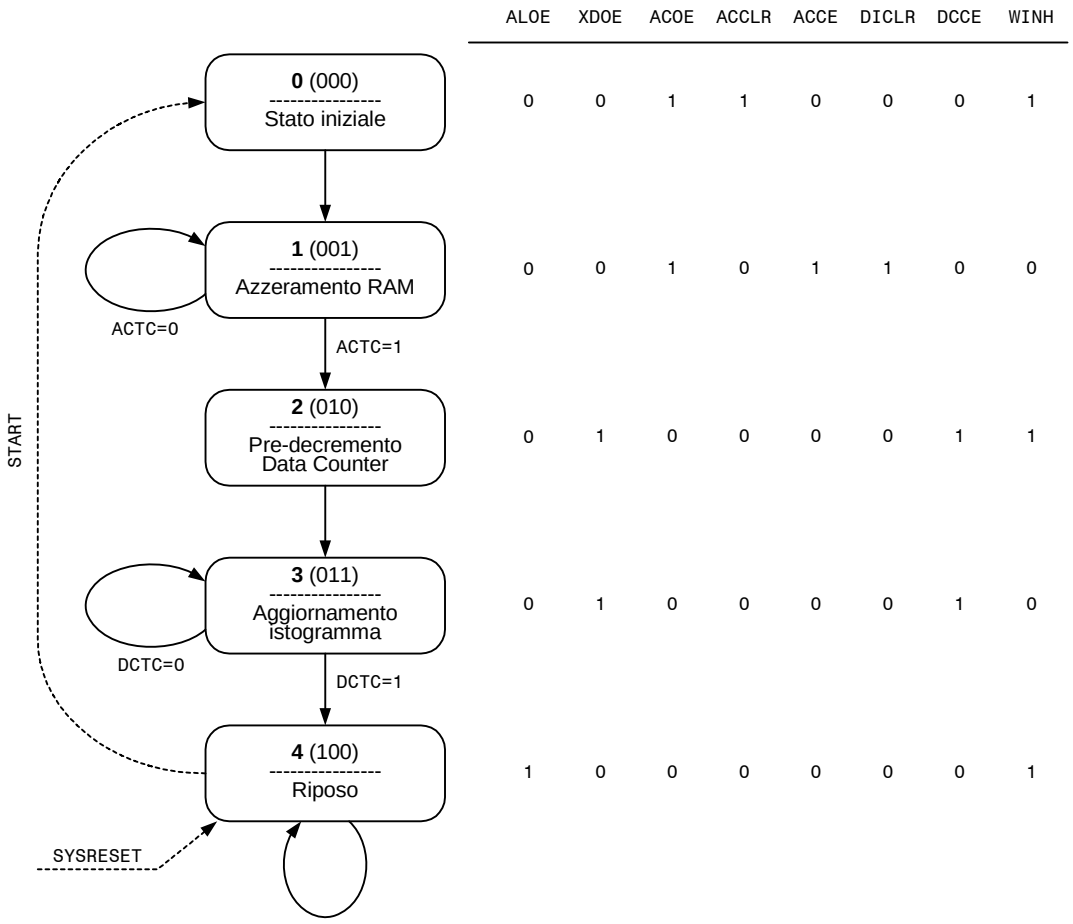


Fig. 10 Schema completo del Sottosistema di Calcolo. Le tre sorgenti per l'indirizzo della RAM (Address Latch, canale dati esterno, Address Counter) sono state dotate di uscita tri-state e inviate in multiplex al latch degli indirizzi di RAM; i tre relativi segnali di Output Enable (ALOE, XDOE, ACOE) saranno gestiti da SCO. Il Data Counter Terminal Count (DCTC) provvede a settare il flip-flop per la richiesta interrupt; tale flip-flop sarà resettato o dall'Interrupt Acknowledge (IACK) oppure dal System Reset (SYSRESET). È anche indicato in basso a destra il circuito per la generazione dei segnali SEL per l'abilitazione delle porte di input/output.



	ALOE	XDOE	ACOE	ACCLR	ACCE	DICLR	DCCE	WINH
0 (000) Stato iniziale	0	0	1	1	0	0	0	1
1 (001) Azzeramento RAM	0	0	1	0	1	1	0	0
2 (010) Pre-decremento Data Counter	0	1	0	0	0	0	1	1
3 (011) Aggiornamento istogramma	0	1	0	0	0	0	1	0
4 (100) Riposo	1	0	0	0	0	0	0	1

Fig. 11 Diagramma di stato del Sottosistema di Controllo, con i valori che devono assumere le variabili di task in ciascuno stato. Il segnale di Start, generato quando la CPU scrive il valore della lunghezza N del blocco, deve forzare lo stato 0 in maniera asincrona, mentre il segnale di System Reset (SYSRESET) deve forzare lo stato 4 di riposo ancora in maniera asincrona. Si noti lo stato 2 (pre-decremento del Data Counter): se N è la lunghezza del blocco, e se il Data Counter emette il Terminal Count al conteggio 0, il primo conteggio utile deve essere $N-1$, diversamente il blocco verrebbe ad essere costituito da $N+1$ dati.

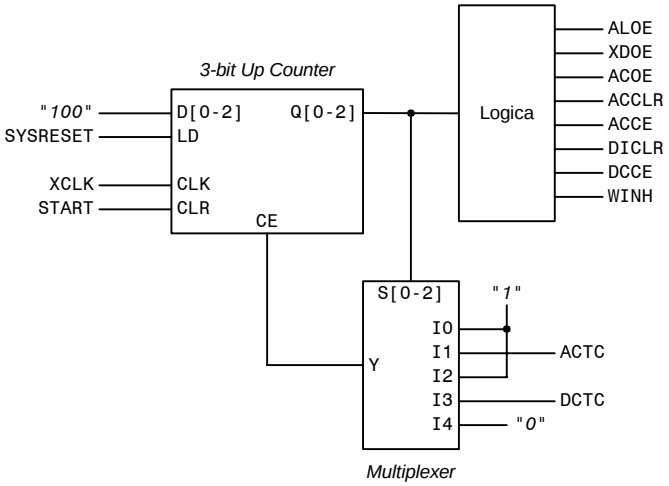


Fig. 12 Implementazione del Sottosistema di Controllo mediante Up Counter a 3 bit.

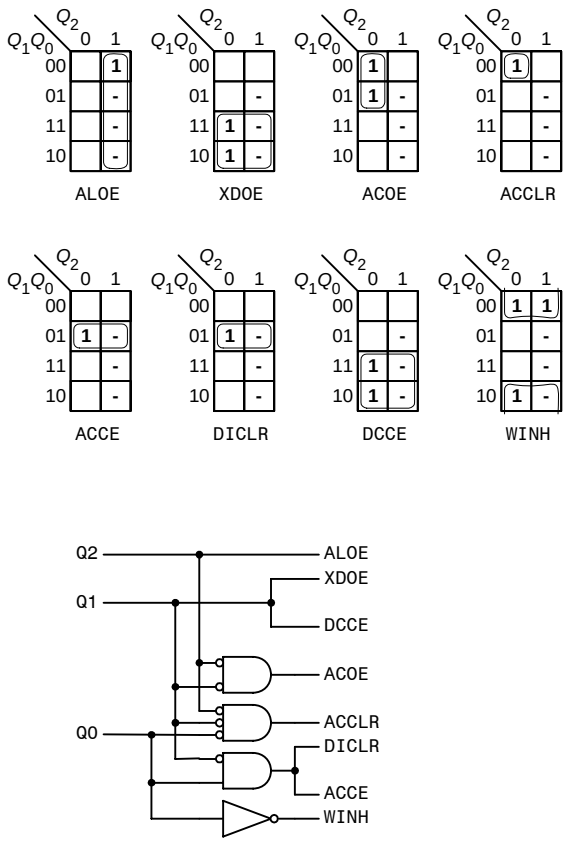


Fig. 13 Logica per la generazione dei segnali di task.

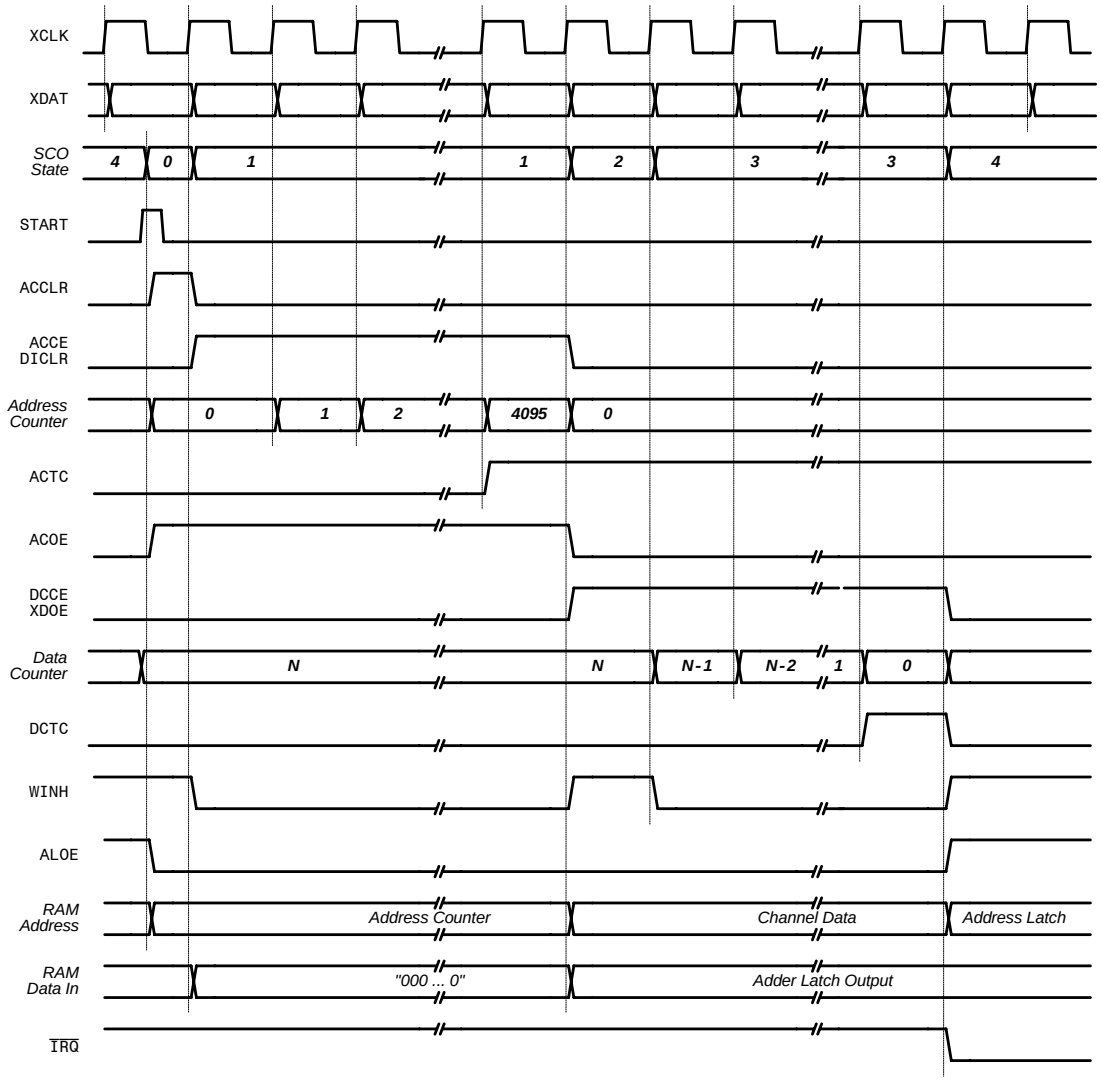


Fig. 14 Temporizzazioni dei segnali di interfaccia.