

**Architettura degli elaboratori I - Modulo B,
Anno Accademico 2001-2002**

Capitolo 6 - Nozioni basilari sui "bus"

1 - Introduzione

La funzionalità di una struttura di elaborazione digitale dipende in misura determinante dalla possibilità che hanno le parti che costituiscono il sistema di scambiare tra loro informazioni con rapidità ed efficienza.

Il caso di collegamento più semplice ma, tutto sommato, anche più frequente è quello in cui sono coinvolte solo due unità, una che trasmette ed una che riceve. Solo di rado succede che dei dati debbano essere inviati da un'unità a due o più altre.

L'invio delle informazioni deve aver luogo con modalità e tempi che vanno stabilite in funzione della struttura interna delle due unità coinvolte nell'operazione.

La prima soluzione che viene in mente per questo problema è quella di stabilire un collegamento diretto dall'una all'altra unità attraverso un cavo di caratteristiche elettriche adeguate. In questo caso si ha quello che nel gergo tecnico si chiama un **collegamento punto a punto**.

La procedura di attivazione del canale di comunicazione prevede di solito che **una delle due unità**, quella che trasmette i dati o quella che li riceve, **prenda l'iniziativa del collegamento** e se le due unità non usano già lo stesso segnale di clock, una delle due deve rinunciare al proprio ed **agganciarsi al clock dell'altra**. Dopo di che, una delle due, non necessariamente quella che ha preso l'iniziativa, si assume il **compito di controllare la quantità delle informazioni e la modalità di trasmissione**. Le informazioni sono, generalmente, parole di un certo numero di bit.

Si tratta di una serie di operazioni che una struttura sequenziale programmabile, come una CPU (Central Processing Unit), effettua con grande efficienza.

Quando, allora, in una struttura digitale esiste una CPU, ragioni di buon senso ed economia consigliano di affidare ad essa il compito di gestire tutti i collegamenti tra unità del sistema, anche quelli in cui essa non è direttamente coinvolta.

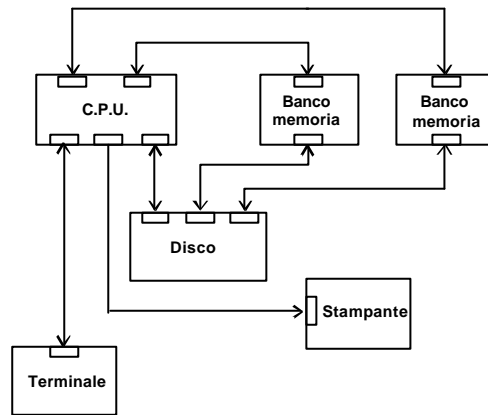


Fig.1

Ciò evita inutili complicazioni in molti dei sottosistemi della struttura portando ad architetture come quella di Fig. 1, in cui è mostrato il tipico schema a blocchi di un computer con **architettura a "stella", intorno alla CPU**. Il concetto che la parola vuole sottolineare è che **tutti gli elementi sono collegati alla CPU**, pur essendo alcuni di essi, anche, collegati tra loro.

E' facile rilevare che una struttura come quella illustrata presenta una **notevole rigidità**, perchè **ciascuno degli elementi del sistema deve incorporare le interfacce verso gli elementi con cui deve interagire**.

Per interfaccia si intende tutto quanto è necessario perché i dispositivi possano scambiare informazioni; si può andare dall'eventuale semplice adattamento dei livelli logici, all'adattamento dei timing, fino a complesse architetture per riorganizzare i dati, nel caso, ad esempio, di una diversa lunghezza della parola nelle due unità.

Si tenga presente che, quando le architetture a stella erano di attualità (anni '50), non era così diffuso come oggi il concetto di standardizzazione e la sostituzione di un periferico con uno non perfettamente uguale, richiedeva un delicato lavoro di adattamento.

Questa tipo di struttura rendeva, anche, il **costo d'acquisto iniziale di un sistema molto alto**, perchè il nucleo base (CPU e memoria), che aveva negli anni '50 le dimensioni di un grosso armadio, doveva contenere in partenza le interfacce verso tutti i possibili periferici di cui era prevista l'utilizzazione.

Il concetto di modularità nei sistemi di elaborazione è in pratica stato introdotto con il "bus", inteso non solo come cavo multifilare che collega l'uno all'altro in catena tutti i componenti di un sistema, ma, anche, come insieme delle regole che bisogna rispettare perchè le informazioni da scambiare fra i sottosistemi siano disponibili al momento giusto e per l'intervallo di tempo necessario per una corretta registrazione.

Questa seconda parte delle specifiche di un bus viene denominata *"protocollo"* e, come vedremo in seguito, ha una importanza fondamentale nel determinare il flusso massimo di informazioni che il bus può convogliare in un certo tempo da un sottosistema ad un altro.

Poichè la cosiddetta *"intelligenza"* di un sistema, intesa come capacità di eseguire operazioni complesse e programmabili, è stata per molti anni concentrata nella sua CPU, era inevitabile che le prime realizzazioni di sistemi di elaborazione modulari, basati su periferici standardizzati, vedessero **la CPU come origine del cavo di connessione ai periferici (bus)** che prevedeva, quando era necessario, una sola terminazione all'altro estremo.

Ciò può apparire oggi come incapacità dei progettisti a cambiare rispetto al passato, in cui, come abbiamo già visto, le interconnessioni punto a punto all'interno di un sistema portavano ad una struttura a stella di cui la CPU era il centro.

Si tratta, invece, di una scelta dettata da esigenze di economia perchè consente di concentrare la gestione del protocollo nella CPU, semplificando al massimo i periferici.

La "rivoluzione Copernicana" nei sistemi digitali fu rappresentata da un sistema di elaborazione in cui la CPU, per la prima volta, era connessa su un bus terminato ad entrambe le estremità.

Essa fu determinata non da motivi filosofici ma, semplicemente, dal fatto che il progresso della tecnologia elettronica aveva nel frattempo reso disponibili circuiti integrati digitali (TTL) compatti ed economici che rendevano possibile dotare anche i periferici della logica necessaria perché essi potessero assumere la gestione del protocollo del bus, senza che questo comportasse costi proibitivi.

La struttura che per prima presentò questa impostazione fu il "*minicomputer*" a 16 bit, **PDP 11/20** della Digital Equipment Corporation (DEC), (Fig. 2) il cui bus venne denominato "*unibus*". Si noti anche che le memorie, che ormai si avviano definitivamente a diventare "elettroniche" (DRAM), sono anch'esse collegate al sistema tramite l'*unibus*.

Si comincia a delineare un primo livello di decentramento logico in vista dell'obiettivo di inserire sullo stesso bus altre CPU.

In realtà, la prima manifestazione di questa tendenza erano state evidenti in un'altra macchina della Digital, il PDP 15, con "parola macchina" a 18 bit, di poco precedente, che pur disponendo di un bus tradizionale, in quanto uscente dalla CPU, prevedeva la possibilità che alcuni periferici fossero in grado di attivare trasferimenti da, o verso, la memoria, senza l'intervento della CPU (**Direct Memory Access, DMA**).

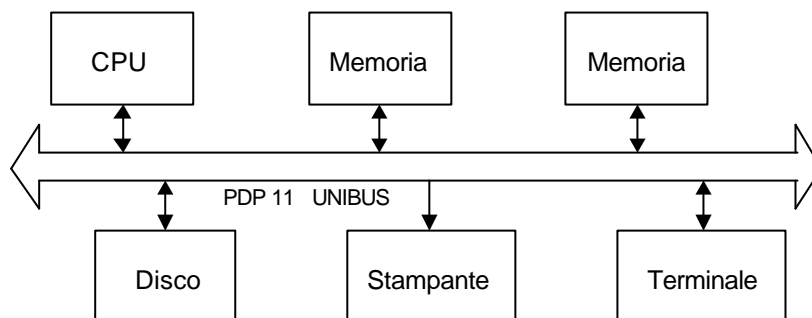


Fig. 2

Sarà bene sottolineare che il bus, come elemento di comunicazione tra sottosistemi, ha almeno un difetto, sacrifica alla modularità la possibilità di parallelizzare i canali di comunicazione tra gli elementi di un sistema.

Naturalmente nei primi anni 70 la cosa appariva perfettamente ragionevole perché si trattava di rinunciare ad una potenzialità solo teorica, dati i costi e gli ingombri della logica di gestione di un protocollo, anche semplice.

Oggi le cose stanno in termini molto diversi ed, infatti, **negli ultimi anni sono stati proposti nuovi standard di comunicazione che utilizzano di nuovo il concetto di collegamento punto a punto, sia pure sovrapposto a quello di bus.**

2 - Evoluzione del concetto di bus

Il tentativo della DEC di trarre il massimo vantaggio dalla sforzo fatto per definire le specifiche dello "unibus" del PDP 11 la costrinse, di fatto, a contribuire notevolmente all'evoluzione del concetto di bus. I suoi progettisti, infatti, nonostante le soluzioni avveniristiche che erano state introdotte nell'*unibus*, si resero ben presto conto che esso, come unico asse portante del sistema, imponeva troppe limitazioni all'evoluzione delle macchine della serie '11 e, già nel modello 11/45 (Fig. 3), affiancarono al bus principale, su cui insistevano logicamente i periferici conformi alle specifiche "unibus", un secondo bus veloce per il collegamento alla CPU dei moduli di memoria "elettronica" (basati su RAM dinamiche), che erano la grande novità tecnologica, rispetto alla memoria principale a nuclei di ferrite, ed un terzo bus (il '*massbus*') appositamente studiato per i dischi magnetici a basso tempo di accesso.

Si faceva strada l'idea di sistema digitale moderno in cui una struttura ad albero, costituita da bus interfacciati l'uno all'altro, assicura i collegamenti necessari tra elementi funzionali modulari.

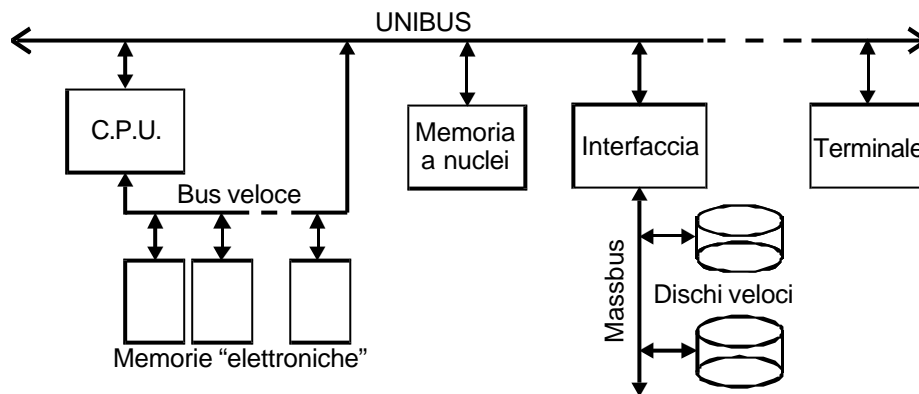


Fig. 3

Presto il concetto di bus sarebbe uscito dall'ambito ristretto dei progettisti di sistemi di elaborazione, per interessare ambienti sempre più vasti di progettisti elettronici.

I cosiddetti '*sistemi di acquisizione dati*', che sono spesso associati ed intrecciati con i '*sistemi di controllo*' e che sono usati in molti settori tecnologicamente avanzati dell'industria e della ricerca, non sono altro che **applicazioni ripetute del concetto di bus.**

I sistemi di acquisizione dati trasferiscono dati in tempo reale da rivelatori o trasduttori ad un sistema di elaborazione (*on line*) che, di solito, dopo aver effettuato una pre-elaborazione ed averli compattati, li trasferisce ad una unità di registrazione. Successivamente essi saranno letti ed analizzati con altri elaboratori (*off-line*).

Anche i sistemi di controllo, acquisiscono dati, ma da trasduttori inseriti in impianti complessi. I valori letti vengono elaborati *in tempo reale* per valutare lo stato di funzionamento ed, eventualmente, intervenire a regolare uno o più parametri

Tutti gli impianti industriali di una certa complessità, da una catena di imbottigliamento di bibite ad una centrale termoelettrica, hanno bisogno di un sistema di controllo.

5 - I bus come elementi costitutivi delle reti di comunicazione digitali

E' evidente che il numero di linee utilizzate da un sistema di comunicazione dipende moltissimo dalle distanze che esso deve coprire. Fin dalla preistoria delle comunicazioni elettriche (telegrafo) è stato chiaro che il costo delle comunicazioni a lineare col numero delle linee utilizzate.

Un sistema che richieda più di una linea bifilare, ha probabilità maggiori di non funzionamento rispetto ad uno che utilizza una sola linea ma, a parità di informazioni per linea nell'unità di tempo, potrà trasmettere un flusso d'informazione maggiore.

In pratica i sistemi multifilari (*bus paralleli*) sono riservati alle basse o medie distanze (da pochi metri a qualche centinaio di metri), i sistemi a poche linee (al limite una) sono utilizzati su lunghe distanze da varie centinaia di metri a molti chilometri.

I primi tendono a trasmettere "*parole*", in genere formate da un numero di bit multiplo di 8, i secondi singoli bit.

E' ovvio che la dizione di bus "*parallelo*" o bus "*seriale*" è in qualche misura opinabile, perchè ha un significato preciso solo quando il dato viene generato con un formato fisso ed il bus ha un numero di linee dati uguale o superiore.

Il bus interno di un sistema di elaborazione, ad esempio, è sempre parallelo con un numero di bit pari a quello della "parola macchina". Ma, se il sistema esegue operazioni aritmetiche in doppia precisione, è evidente che l'informazione elementare avrà un numero di bit doppio rispetto al bus, se questo ha le linee dati in numero pari ai bit della parola, e dovrà essere trasmessa su di esso attraverso una operazione di multiplexing in due parole successive.

Naturalmente anche un bus ad una sola linea può trasmettere flussi di informazioni molto alti se la massima frequenza di trasmissione è molto elevata. Si pensi ad esempio ai moderni standard di trasmissione su fibra ottica.

I bus paralleli dei moderni computers sono a 32, 64 o 128 bit, ma il numero di linee utilizzato è notevolmente superiore perchè sono necessarie, oltre alle *linee dati*, le linee per gli indirizzi ed altre linee per i *segnali di controllo e temporizzazione* che il protocollo di comunicazione richiede e che, inevitabilmente, crescono con le esigenze di elevati flussi di informazione.

Ciò spiega l'affermazione fatta sopra che i bus paralleli sono utilizzati solo su brevi distanze. Il costo e l'ingombro dei relativi cavi sono una grossa difficoltà per utilizzazioni più estese.

Su questa base nei primi anni 70 sono nate le reti di comunicazione digitali (Network) in cui un certo numero di computers di grande potenza, ciascuno connesso ai suoi periferici con bus parallelo, erano in comunicazione tra loro con linee seriali a velocità medio basse. Si è cominciato con decine di Kbit/s ora siamo a molti Mbit/s.

Le reti a seconda della loro estensione sono dette LAN (Local Area Network), MAN (Metropolitan Area Network) o WAN (Wide Area Network).

Le reti di computer sono nate come evoluzione dei collegamenti su linea telefonica che, a loro volta, erano nati per consentire ad utenti remoti di utilizzare per tempi limitati le risorse di

calcolo dei “*supercomputer*” detti anche “*mainframe*” dei “*centri di elaborazione*” commerciali o scientifici che nascevano alla fine degli anni sessanta.

Gli utenti inviavano i dati che volevano elaborare alla sede del centro, registrati su nastro magnetico, poi si collegavano da un terminale remoto per scrivere od editare il loro programma, per poi mandarlo in esecuzione quando ricevevano per telefono assicurazione dall’operatore del centro di elaborazione che il bro nastro dati era stato messo *in linea* (montato su una unità di lettura di nastri magnetici). In questo modo sulle linee viaggiavano solo poche informazioni relative al programma ed a velocità ridotta.

Successivamente la disponibilità, a prezzi sempre più bassi, di piccoli o medi sistemi ha consentito di ottimizzare i collegamenti col supercomputer facendoli gestire dal sistema locale e sono nate le reti di computer vere e proprie, ma esse sono state per quasi due decenni sistemi di comunicazione, a distanze medio grandi, tra sistemi di elaborazione concentrati.

L'evoluzione della tecnologia delle trasmissioni ad alta velocità su linee seriali le sta, sempre più, trasformando in sistemi di elaborazione distribuita in cui l'elaborazione può avvenire in un qualsiasi nodo della rete, od addirittura in più nodi, senza che i tempi di trasmissione dei dati e dei programmi abbiano un peso determinante.

Grazie ai contemporanei progressi dei sistemi operativi, l'elaborazione distribuita appare completamente trasparente al singolo utente che non ha bisogno di intervenire nell'assegnazione dell'elaborazione (job) che il sistema operativo fa automaticamente al nodo meno impegnato.

I sistemi con molte CPU, in grado di lavorare in parallelo su un unico programma (i cosiddetti “*server*”) hanno, nel frattempo trasformato il concetto di supercomputer. **Quello che una volta si otteneva da una CPU basata su tecnologie molto più avanzate di quelle dei piccoli sistemi, oggi si ottiene con un certo numero di processori praticamente uguali a quelli dei piccoli sistemi che, grazie ai progressi del software e di sofisticatissime architetture sistemiche basate su bus paralleli superveloci, concorrono ad assicurare elevate prestazioni.**

6 - Elementi costitutivi e terminologia dei bus.

Le linee di un bus possono essere classificate in base alla loro specifica funzione.

Abbiamo già visto che esistono le *linee dati* che sono in fondo la prima motivazione di un bus. E' ovvio che debbono esistere delle *linee indirizzo* per controllare la sorgente o la destinazione dei dati. E' altrettanto ovvio che il numero delle linee indirizzo determina il numero massimo di dispositivi che si possono connettere al bus.

Vi sono, poi, le *linee di temporizzazione* che, come vedremo, insieme alle *linee di controllo* hanno un ruolo importantissimo nella gestione del protocollo.

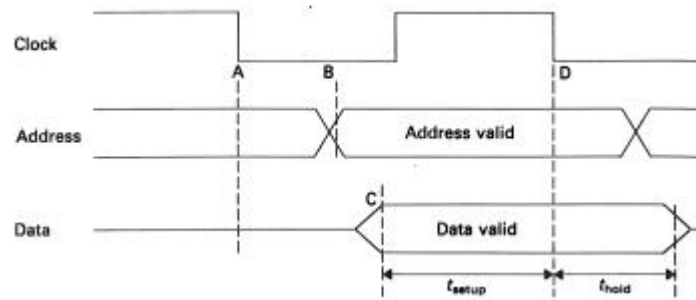


Figura 4

Il protocollo può essere di *tipo sincrono* (figura 4) se è consegnato in modo che, una volta avviata la trasmissione dei dati, essa avvenga con cadenza regolare e fissa (N bit per periodo del clock) o di tipo *asincrono* (figura 5) nel senso che la frequenza di trasmissione dei dati si adegua automaticamente alle caratteristiche del dispositivo che trasmette e di quello che riceve.

Si tenga presente che spesso nel gergo si usa parlare di *bus dati*, *bus indirizzo* o *bus di timing e controllo* si attribuisce, cioè, la qualifica di bus ad ogni gruppo di linee con una funzione diversa.

E' molto importante tener presente che i quattro tipi di linee che esistono sempre, con numeri diversi, in tutti i bus paralleli, corrispondono a funzionalità che sono indispensabili anche nei bus seriali.

E' ovvio che, quando il numero fisico di linee deve essere ridotto, è necessario che le funzionalità corrispondenti alle linee che non possono esistere fisicamente, debbono essere svolte da segnali che in determinati istanti viaggiano su quelle esistenti.

Si ricorre, cioè, alla tecnica del multiplexing di più tipi di segnale su una stessa linea od anche, come si dice in gergo, al *time sharing* delle linee a disposizione tra le diverse funzioni.

Molti bus paralleli utilizzano il multiplexing sulle stesse linee fisiche di indirizzi e dati. Basta pensarci un attimo, per rendersi conto che la cosa è tutt'altro che stupida, visto che nella fase di impostazione della trasmissione dei dati, in cui funzionano le linee indirizzo, le linee dati sono necessariamente inutilizzate.

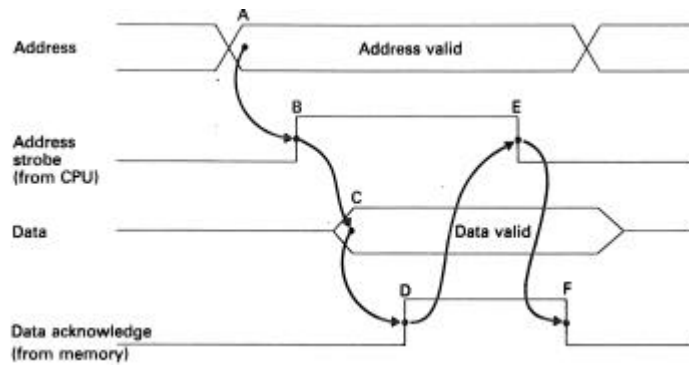


Figura 5

E' ovvio che si tratta di scelte economicamente vantaggiose ma che vanno a discapito della velocità di trasmissione (*flusso dati*) e complicano notevolmente il protocollo.

La tecnica del multiplexing celebra il suo trionfo nei bus seriali, in particolare in quelli monolinea.

In questo caso, se si utilizza un protocollo sincrono, ogni tipo di segnale deve avere una ben determinata fase rispetto al segnale di temporizzazione. Se, invece, si utilizza un protocollo asincrono ogni tipo di segnale deve essere preceduto da una assegnata sequenza di segnali che, opportunamente decodificata dalla stazione ricevente, consentano di riconoscere i bit dati da quelli indirizzo e questi dai segnali di controllo.

I dispositivi collegati ad un moderno bus possono essere di due tipi: *master* o *slave*.

I primi sono quei dispositivi che possono assumere la gestione del protocollo del bus.

Si tenga presente che questo non significa necessariamente che il master abbia l'esclusiva dell'iniziativa, perché **anche uno slave (dispositivo periferico) può richiedere il collegamento per scambio di informazioni.**

Si tenga, anche, presente che un *master* non deve necessariamente essere una CPU, cioè non deve essere per forza "*intelligente*", nel senso già visto. E' sufficiente che abbia **produrre i segnali di timing e di controllo previsti dal protocollo, nella giusta sequenza.**

Se il bus ammette più di un master, allora solo uno di essi può essere attivo.

Un solo master per volta può gestire il protocollo del bus ed in questo caso esso si chiama *commander*.

Il commander di un bus esegue transazioni che possono coinvolgere anche più di uno slave.

Quando una transazione interessa tutti gli slave si dice di tipo broadcast.

E' opportuno precisare che un master non attivo può comportarsi da slave e, quindi, ricevere o trasmettere dati sotto il controllo del commander.

Un bus multi-master comporta automaticamente il problema dell'*arbitraggio del bus* cioè deve avere una struttura hardware ed un protocollo idoneo a fronteggiare la situazione che si viene a

creare quando il master che era attivo ha terminato le sue transazioni e, nel frattempo, due o più degli altri master hanno attivato la linea di richiesta del controllo del bus, candidandosi al ruolo di commander.

Vedremo, in seguito che **questo problema ha molte analogie con la situazione che si determina quando più slave rivolgono contemporaneamente richiesta di servizio al commander del bus (*program interrupt*)**.

Molte delle tecniche che si possono adoperare per gestire le priorità di collegamento si possono adoperare per arbitrare le richieste di controllo del bus.

La richiesta di servizio che uno slave rivolge al commander del bus comporta, se quest'ultimo è un dispositivo che lavora in maniera asincrona rispetto a quello richiedente, il **problema della sincronizzazione. Lo slave deve, cioè, agganciarsi al clock del master che farà da riferimento per i segnali di temporizzazione del protocollo del bus.**

Se il commander è, poi, un dispositivo programmabile (intelligente), esso potrebbe essere impegnato ad eseguire un programma (*main program*).

L'effettuazione della transazione richiesta è generalmente incompatibile con altre quindi richiede che il dispositivo interrompa (nei modi e nei tempi opportuni) il programma in corso (*program interrupt*), individui il dispositivo od i dispositivi richiedenti, selezioni quello a più alta priorità per poi sincronizzarsi con esso per la trasmissione dati.

Quest'ultima avverrà, di solito, eseguendo un apposito programma che costituisce la *routine di servizio* del periferico in questione.

Nel caso il commander non sia intelligente, ma sia un *hardware processor*, si procederà direttamente alla sincronizzazione ed allo scambio dati, in applicazione di un protocollo necessariamente più rigido.

3 - Il bus di un microprocessore della seconda generazione

L'architettura di un calcolatore della attuale generazione che qualcuno definisce quinta, qualcun altro sesta, non è la più adatta a far capire quali sono i principi essenziali su cui essa poggia.

Una architettura storicamente rilevante o, come nel caso della "macchina di Mano", appositamente concepita, risulta didatticamente più efficace.

Lo stesso discorso vale per il bus, che è una delle strutture portanti di una moderna architettura di computer. Il ricorso ad un sistema della seconda generazione di CPU single chip, quella successiva alla prima, indissolubilmente legata all'8080 della I opportuna.

Il microprocessore a cui ci riferiremo, il CDP 1802 è un microprocessore single chip in tecnologia CMOS, è stato uno dei primi con una sola alimentazione e, per giunta, compatibile con i 5 Volt dei circuiti integrati TTL. Ha avuto un successo sistemistico molto scarso, ma si presta splendidamente per dare una idea abbastanza semplice ma completa di un bus interno di un processore

Gli attuali potentissimi microprocessori single chip sono talmente complessi che sarebbero totalmente inutilizzabili se non nascessero già corredati di una serie impressionante di strumenti hardware (sotto forma di circuiti integrati VLSI per le funzioni accessorie) e

software, appositamente predisposti per gli usi più ricorrenti. Questi strumenti, inevitabilmente, lasciano pochissimo spazio alla fantasia del singolo progettista a livello di bus.

Il CDP 1802, invece, quando apparve sul mercato, alla metà degli anni settanta, era un prodotto tecnologicamente di punta (CMOS), dovuto ad una ditta emergente, la RCA, oggi completamente fuori dal mercato dei processori, che cercava una strada autonoma nel mondo dei produttori di CPU e, mentre presentava una struttura del processore molto originale, proponeva un bus molto semplice e duttile, perfettamente in linea con gli sviluppi che queste strutture hanno poi avuto negli anni successivi.

Naturalmente, proprio per questa sua duttilità, **il bus del CDP 1802 è estremamente utile in chiave didattica**. Esso si presta a fornire gli elementi costitutivi per vari, plausibili, scenari interpretativi che possono servire a chiarire vantaggi e svantaggi di certe strutture dei bus.

In queste dispense non si parlerà della struttura interna della CPU della RCA che come accennato, è molto particolare, e non si parlerà, se non indirettamente, del set di istruzioni assembler del dispositivo.

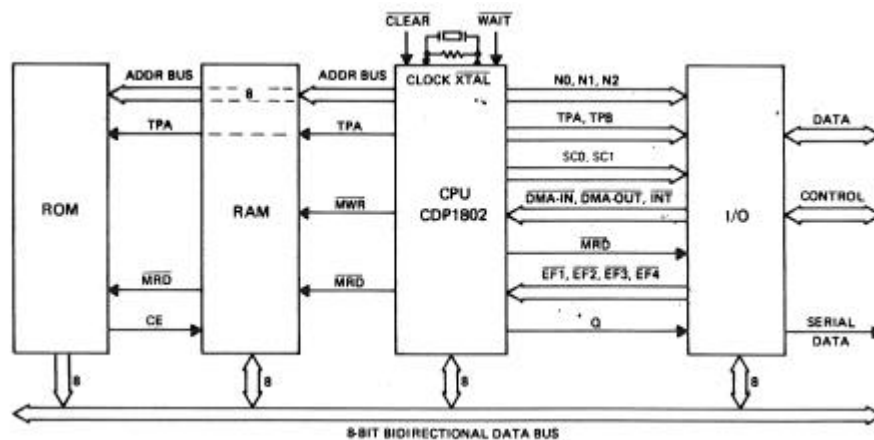


Fig. 6

Nella figura 4 è mostrato lo schema di un sistema basato su un CDP1802.

Il sistema è costituito oltre che dalla CPU, dalla memoria, in parte a sola lettura per la parte fissa del software (monitor) ed in parte a lettura e scrittura, per i programmi applicativi ed i dati e dal processore di "Input/Output" (I/O). Quest'ultimo, oltre ad essere collegato, come gli altri sottosistemi, al bus dati, gestisce tutti gli altri pin della CPU.

Le linee a disposizione del progettista del sistema dei periferici sono:

- 8 linee di indirizzamento,
- 8 linee dati (bidirezionali),
- 2 linee di temporizzazione TPA e TPB,
- 2 linee di gestione memoria MRD* ed MWR*.

- (e) 4 linee per attivazione flag, EF1*, EF2*, EF3*, EF4*;
- (f) 3 linee per richieste di I/O, DMA-IN*, DMA-OUT*, INT*;
- (g) 3 linee di indirizzamento periferici, N1, N2, N3,
- (h) 1 linea controllabile in software, Q
- (i) 2 linee di status della CPU, SC0, SC1.

Le linee con asterisco operano in logica negativa. Tutte le linee mono-direzionali entranti sono in logica negativa (lo stato attivo è il livello basso) perchè esse costituiscono i canali che i periferici hanno a disposizione per collegarsi con la CPU e la logica negativa è indispensabile, er realizzare, se necessario, un wired-or fra più dispositivi.

I nomi delle linee contengono i suggerimenti che il costruttore da al progettista per il loro uso.

Per le linee del punto (f), ad esempio, le prime due sono chiaramente previste per le richieste di connessione diretta alla memoria per scriverci dati, la prima, e per leggerli, la seconda.

La terza linea (INT*) serve per le richieste di "program interrupt", operazione indispensabile per attivare trasferimenti dati che utilizzano la CPU.

Le reazioni della CPU all'attivazione di ciascuna delle tre linee sono previste nella struttura hardware del dispositivo e, per esempio, l'attivazione della linea di DMA-IN* produce automaticamente la scrittura in memoria, all'indirizzo presente sulle apposite linee, del dato presente sulle linee dati, senza che la cosa interessi i registri interni della CPU, la cui attività viene solo rallentata per il tempo che è necessario per eseguire il ciclo di registrazione.

Analogamente avviene per il ciclo di DMA-OUT*, in cui il dato presente alla locazione indicata da un apposito registro puntatore interno alla CPU e che può essere caricato da software, viene messa sulle linee dati. In caso di nuova attivazione della linea il puntatore viene automaticamente aggiornato alla locazione successiva della memoria e così via per il numero di bytes di cui il dispositivo periferico ha bisogno.

Il DMA IN ed il DMA OUT del RCA CDP1802 sono interpretazioni particolari del concetto di DMA.

In pratica essi sono "facilitati" dalla CPU, per rendere più semplice la logica interna dei periferici. Che, altrimenti avrebbero dovuto essere in grado di gestire autonomamente il *protocollo* del bus.

Nei moderni sistemi, come vedremo ad esempio, nel 68.000 della Motorola, la CPU mette le sue linee dati nello stato di "alta impedenza ed il compito di gestire indirizzi e dati è lasciato completamente al periferico.

La linea di INT*, invece, **produce il blocco dell'attività della CPU**, alla fine del ciclo istruzione in corso, ed il salvataggio del contenuto dei registri interni essenziali della CPU (il "*program counter*", il registro indice e il contenuto dell'accumulatore) in apposite locazioni di memoria.

Tutto ciò avviene perché, come nella macchina di Mano, ad un certo punto del normale ciclo di esecuzione dell'istruzione corrente, la logica di controllo va a "*testare*" lo stato logico della linea INT*. Quando la trova attiva, effettua i salvataggi di cui abbiamo parlato e, poi, carica nel *program counter* l'indirizzo di una predeterminata locazione di memoria che contiene la prima istruzione della **routine di individuazione del periferico richiedente** se è stato impostato un ciclo di indirizzamento "*diretto*" o contiene l'indirizzo della locazione dove la routine comincia, se nel progetto della logica di controllo è stato impostato un indirizzamento "*indiretto*".

Naturalmente il programma da eseguire deve corrispondere ad una precisa struttura hardware. Una possibilità è, ad esempio, che si utilizzi una linea uscente (potrebbe essere la Q) per stabilire un ciclo di "*hand-shaking*" col sistema dei periferici.

Se si desidera limitare al minimo l'hardware si può usare la struttura detta "*daisy chain*", illustrata in figura 5. Si tratta di una procedura asincrona in cui viene utilizzata il wired-OR sulla linea di INT* per le richieste e la linea di Q (in logica positiva) per l'abilitazione a procedere (GRANT) ad interruzione avvenuta. Il segmento trasversale sul simbolo dei driver NOT dello schema ci ricorda che sono del tipo "*open drain*".

E' la prima istruzione della routine di *Program Interrupt* a porre Q ad "1". Il segnale di GRANT viene "*trattenuto*" (non propagato) dal primo dei dispositivi della catena che ha il FlipFlop di "*Flag*" attivo. A questo punto il dispositivo mette sulle linee dati il proprio codice identificativo e lo "*conferma*" portando la linea di EF1* bassa (*handshake*).

Poiché il codice del dispositivo è detto anche "*vettore*", si parla di procedura di *interrupt vettorizzato*.

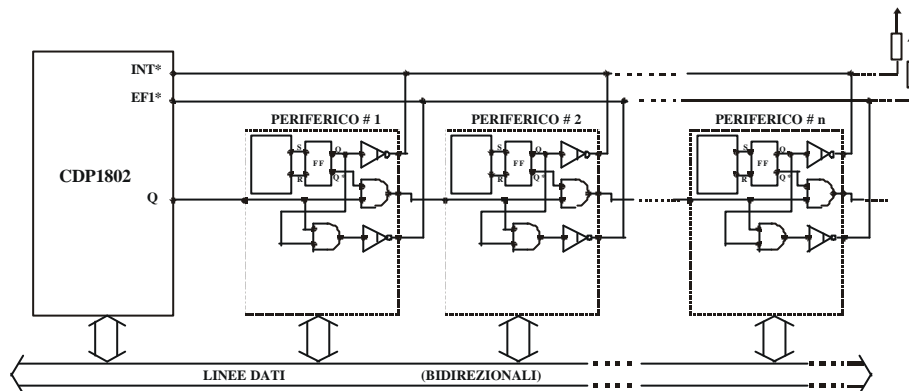


Fig. 7 Hardware per la daisy chain.

La linea EF1* bassa avverte la CPU che il codice è pronto per essere registrato. Il codice, di solito, è l'indirizzo di una locazione della pagina 0 di memoria dove è registrato l'indirizzo iniziale del "*driver software*" del periferico.

L'esecuzione della routine di servizio, scritta nel linguaggio assembler del processore ed in funzione delle caratteristiche del periferico, trasferirà i dati, di cui il periferico ha bisogno, prelevandoli dalle locazioni della memoria del processore che il progettista del sistema ha associato al dispositivo o, viceversa, registrerà in memoria i dati che il periferico fornirà, disponendoli a partire da una assegnata locazione della memoria.

La temporizzazione dei segnali sul bus è quella di figura 7 bis.

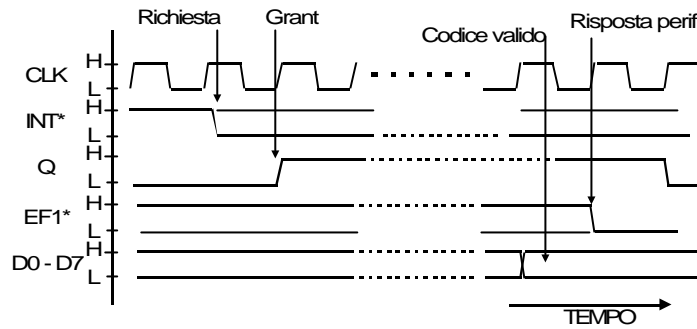


Figura 7 bis

La *daisy chain* assegna la priorità più alta al dispositivo più vicino alla CPU nella catena.

Per quanto riguarda l'uscita dall'interruzione si può decidere che le *routine di servizio* di tutti i periferici finiscano con un gruppo di istruzioni che ripristinino il contenuto dei registri principali per consentire la ripresa del programma in corso, oppure che le routine dei periferici rinviino il controllo alla *routine di interrupt* che provvede ad effettuare le operazioni necessarie per la ripresa del programma che era in esecuzione al momento dell'interruzione (*main program*).

Una alternativa possibile per la procedura di individuazione del periferico richiedente l'interruzione può essere quella in cui la CPU, **dopo che è stata eseguita la parte di programma relativa al salvataggio dei registri, come nel caso precedente**, seleziona in successione i dispositivi collegati alla linea INT*, utilizzando le linee N0, N1, N2, (codificate se i dispositivi sono più di 3).

In questo caso (**tecnica del polling**) la priorità viene assegnata in software dall'ordine con cui sono stati inseriti i dispositivi nella routine di servizio

Si tenga presente che se le linee N sono usate codificate i dispositivi possono essere 8 e, se non si vuole sacrificare una delle combinazioni (per esempio 000) sarà necessario confermare l'indirizzo con un segnale di consenso, per il quale si può usare la stessa linea Q, ma, in questo caso, il livello alto deve viaggiare su una semplice linea uscente del tipo "*omnibus*". Che sia collegata, cioè, a tutti i dispositivi. La struttura dell'hardware che i periferici debbono contenere è schematizzata in figura 8.

program interrupt”, la cui struttura è questo tipo:

```
.....  
I      porta bassa la linea Q  
I + 1  scrivi il codice Y sulle linee N, ----- Inizio primo ciclo  
I + 2  porta alta la linea Q,  
I + 3  pausa  
I + 4  la linea EF1* è bassa? se si, esegui la prossima istruzione, se no, saltala,  
I + 5  trasferisciti all'indirizzo XX (dove comincia la routine del periferico Y).  
I + 6  porta bassa la linea Q, ----- Inizio ciclo successivo.  
I + 7  Scrivi il codice Z sulle linee N  
I + 8  Porta alta la linea Q,  
I + 9  .....
```

.....
L'esempio di routine è relativo al caso in cui si desidera esplorare i dispositivi secondo un ordine qualsiasi. Se semplicemente si vogliono interrogare i periferici secondo l'ordine naturale dei codici binari da 0 a 7, basta programmare un ciclo software indicizzato su un registro usato come contatore binario.

La corrispondente temporizzazione dei segnali sul bus è riportata in figura 8 bis in cui è mostrato il caso in cui il periferico, avendo richiesto l'interruzione, risponde. Negli altri casi dopo un tempo opportuno (time out) si procede ad interrogare il successivo periferico.

Vorrei sottolineare che l'uso del segnale Q per confermare l'indirizzo oltre che consentire l'utilizzazione di tutte le combinazioni possibili delle 3 linee (8) è in realtà fondamentale perchè, contrariamente a quanto schematizzato in fig. 8, è **consigliabile che la convalida dell'indirizzo avvenga con dispositivi "edge triggered" perchè in questo modo si minimizza il rischio che un segnale spurio possa determinare un errore di selezione.**

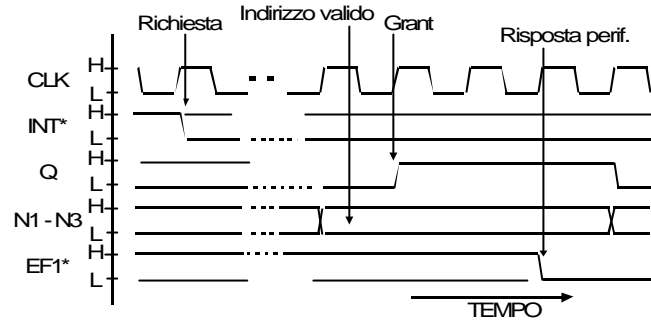


Fig. 8 bis Temporizzazione del polling

Negli schemi di principio di figura 8 non compare il segnale di clock per non complicare le strutture logiche ma il clock è fondamentale perché le strutture che interagiscono tra loro attraverso un bus asincrono, come quello ipotizzato sono, in generale, macchine sequenziali sincrone e durante le *transazioni* sul bus debbono usare lo stesso clock.

E' forse opportuno precisare che la routine di servizio del periferico che viene messa in esecuzione dopo la risposta sulla linea viene individuata usando il codice messo sulle linee N, come al solito attraverso un indirizzamento diretto o, più spesso indiretto alla corrispondente locazione di memoria..

E' utile osservare che nel caso di un sistema semplice come quello del CDP 1802 si da per scontato che **il bus di sistema sia unico**, nel senso che gli 8 bit dati, ad esempio, viaggino in parte su linee interne al chip del μP ed in parte su fili di un cavo che esce dal nucleo fisico essenziale del sistema, costituito dalla CPU e dalla memoria, che, di solito, sono su un circuito stampato.

Questo tipo di soluzione è stato completamente abbandonata nei sistemi moderni perché costringe ad usare per le transazioni tempi dettati dai segmenti del bus più lenti (elevate capacità parassite) e più lunghi.

In linea di massima, un bus più è corto e più è veloce. E' noto, infatti che il bus di sistema sulle motherboard dei personal computer, il PCI ha una lunghezza di pochi centimetri.

3 - Il bus del microprocessore MC 68.000.

Questo microprocessore della Motorola è stato il capostipite di una numerosa e gloriosa famiglia di CPU.

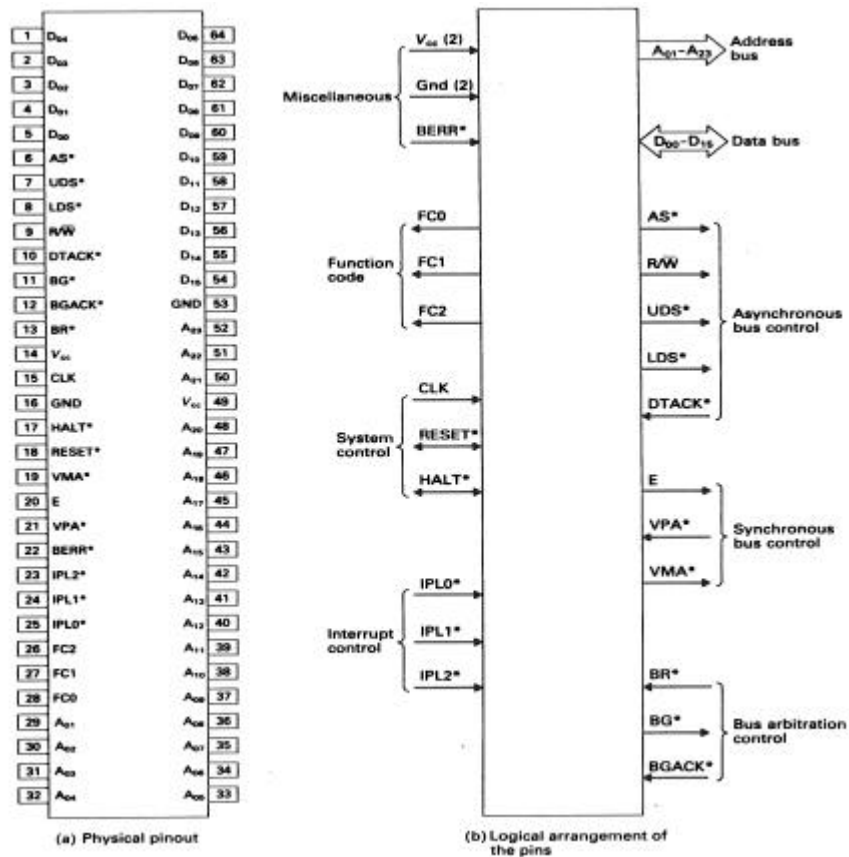


Fig. 9 – Utilizzazione dei pin del μ P MC68.000

Noi ci riferiremo al bus del 68.000 perché contiene tutti gli elementi base ed è meglio documentato, anche perché è l'unico che sia stato usato anche come bus esterno e ciò è avvenuto, per giunta, nel famoso esperimento di Fisica Subnucleare i cui risultati hanno fruttato il premio Nobel al Prof. C. Rubbia (UA1 al CERN).

Di solito, infatti, i bus interni delle CPU sono ottimizzati per operare con pochi dispositivi e su brevi distanze.

Vedremo più avanti che per il collegamento dei periferici, lo stesso costruttore del microprocessore offre chip VLSI ottimizzati per un bus esterno, in grado di sfruttare a fondo il set d'istruzioni, senza che l'allungamento delle linee ed il traffico di informazioni dei periferici rallenti gli accessi della CPU alla memoria centrale.

La utilizzazione dei 64 pin del componente MC 68.000 è illustrata in figura 9.

In base a quanto detto finora sui meccanismi di funzionamento delle tipiche strutture logiche dei bus, la semplice esplicitazione delle sigle associate ai piedini del dispositivo presentata in tabella 1 dovrebbe essere sufficiente per farsi un'idea della struttura del bus e delle sue potenzialità. Naturalmente, il quadro del bus per poter essere esauriente dovrebbe contenere anche le specifiche temporali complete del protocollo e questo richiederebbe una notevole serie di diagrammi temporali e, discorsi approfonditi sulla struttura interna della CPU e sull'uso delle istruzioni assembler associate.

Ci limiteremo a fare solo qualche esempio e riferito, in particolare, agli accessi della CPU in memoria ma questo non riduce l'interesse dell'informazione anche perché quando si usa la tecnica del "memory mapping" (come nel sistema MacUA1) tutti i periferici vengono visti come parti di memoria.

Esistono tutta una serie di chip VLSI, appositamente progettati per l'interfacciamento delle CPU della serie 68k ad un sistema di periferici. Questi chip sono in grado di soddisfare tutte le possibili esigenze dell'utente ed alcuni sono pensati per fare dei dispositivi della famiglia 68k la soluzione ideale per il bus standard VME.

Il 68.000 è stato, tra l'altro, progettato in modo da essere anche compatibile, come vedremo, coi dispositivi d'interfaccia del suo predecessore in casa Motorola, il 6.800, che aveva un bus sincrono.

Le linee di alimentazione, massa, clock, reset ed halt non hanno una specifica rilevanza per la connessione alla memoria ed ai periferici. Le 23 linee indirizzo e le 16 linee dati sono, invece, fondamentali per lo scambio di informazioni con la memoria ed i periferici. I pins **A01 A23** permettono l'indirizzamento diretto ad una memoria di 2^{23} (8 M) locazioni da 16 bit corrispondenti a 16 MByte.

Essi, come si vede dalla tabella 1 sono, per la CPU, uscite, ma hanno dei driver tristate perché, in determinate condizioni, le corrispondenti linee possano essere gestite anche da un periferico.

Le linee **D_{xx}** sono bidirezionali e possono essere messe nello stato di alta impedenza per consentire ad altri dispositivi di prenderne il controllo. Quando vengono effettuati trasferimenti di un solo byte possono essere adoperate solo le linee da **D₀₀** a **D₀₇** oppure quelle da **D₀₈** a **D₁₅**. Durante un ciclo di accettazione di interruzione le linee da **D₀₀** a **D₀₇** possono essere usate dal dispositivo che ha richiesto l'interruzione per fornire alla CPU il proprio codice ("vettore"). E' ovvio che questo pone a 256 il limite superiore al numero di periferici che possono essere collegati ad un sistema di interrupt vettorizzato.

Le linee indirizzi e dati operano in stretta connessione con cinque segnali di controllo: **AS***, **UDS***, **LDS***, **R / W*** e **DTACK*** che in opportuna combinazione consentono la gestione dei trasferimenti dati tra la CPU e la memoria esterna.

I pin **A₀₁**, **A₀₂**, **A₀₃** hanno una funzione specifica nell'interfacciamento coi periferici. Essi, infatti, durante le interruzioni sono usati dalla CPU per comunicare al sistema il codice del livello di priorità che essa sta servendo.

AS*: E' un segnale attivo basso come tutti quelli con l'asterisco. La sua attivazione significa che l'indirizzo sulle linee A è valido.

Power supply	V _{cc}	—	—
Ground	GND	—	—
Clock	CLK	Input	—
Reset	RESET*	I/O	OD
Halt	HALT*	I/O	OD
Address bus	A ₀₁ -A ₁₃	Output	TS
Data bus	D ₀₀ -D ₁₃	I/O	TS
Address strobe	AS*	Output	TS
Read/write	R/W	Output	TS
Upper data strobe	UDS*	Output	TS
Lower data strobe	LDS*	Output	TS
Data transfer acknowledge	DTACK*	Input	—
Bus error	BERR*	Input	—
Enable	E	Output	TP
Valid memory address	VMA*	Output	TS
Valid peripheral address	VPA*	Input	—
Bus request	BR*	Input	—
Bus grant	BG*	Output	TP
Bus grant acknowledge	BGACK*	Input	—
Function code output	FC0, FC1, FC2	Output	TS
Interrupt priority level	IPL0*, IPL1*, IPL2*	Input	—

TS = tristate output	Input = input to the 68000
TP = totem-pole output	Output = output from the 68000
OD = open-drain output	I/O = input or output

Tab. 1 - Utilizzazione e struttura dei pin

R / W*: Quando è in corso un ciclo di accesso alla memoria se questo pin è ad uno stato logico alto la CPU sta leggendo un dato in memoria se, invece esso è basso la CPU sta scrivendo un dato in memoria. Quando non c'è accesso alla memoria e la CPU sta facendo una qualsiasi operazione interna la linea è mantenuta sempre alta. Quando però la CPU entra in una delle condizioni che portano le sue linee controllo in alta impedenza la linea rimane floating ed, allora, bisogna che qualche altro dispositivo od un resistore di "pull up" impediscano alla linea di andare bassa, abilitando la scrittura in memoria di dati erronei.

UDS* ed **LDS***: Sono i segnali di validazione del byte superiore (D_8, D_{15}) e del byte inferiore (D_0, D_7) del bus dati. Quando il dato in trasferimento è una parola (16 bit) i due segnali di "strobe" vengono attivati contemporaneamente.

DTACK*: E' il segnale che conclude il ciclo di "handshake" per il trasferimento di un dato. Esso deve essere generato dal dispositivo interlocutore ed avverte la CPU che il dato sulle linee è valido. Una volta avviato un trasferimento, se questo segnale non va basso la CPU rimane bloccata, pertanto, di solito, si predispone un circuito di temporizzazione che, dopo un tempo prestabilito, genera comunque un **DTACK***, associato, però, ad un segnale di errore, per segnalare che il dispositivo interlocutore (es. la memoria) non ha generato il segnale nel tempo limite.

Si noti la terminologia molto sottile che assegna solo ad alcuni segnali di validazione la qualifica di "strobe". Si tratta di segnali il cui contenuto d'informazione è concentrato nella transizione e, quindi vanno trattati solo con logiche "edge triggered". Naturalmente, anche gli altri possono essere trattati con funzioni sensibili alla transizione.

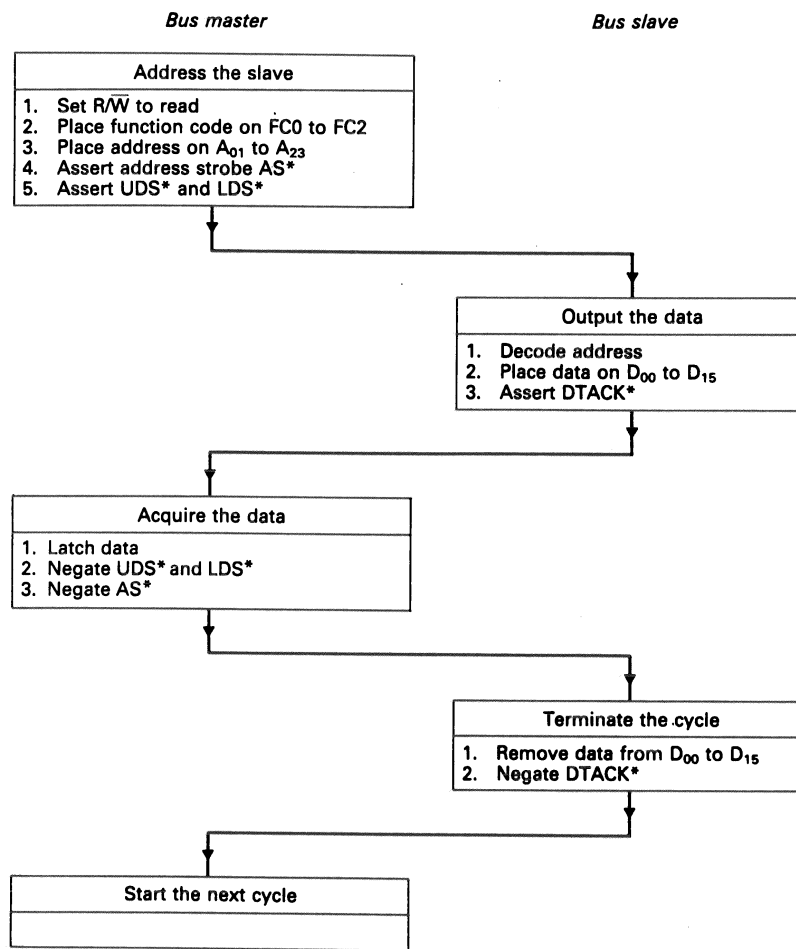


Fig. 10 - Diagramma delle operazioni di un ciclo di read sul bus

I pin la cui funzione è stata discussa finora sono quelli che sono usati anche per gli accessi in memoria e, quindi, sono indispensabili in qualsiasi applicazione della CPU.

A titolo di esempio, nel seguito è riportato il diagramma di flusso di un ciclo di lettura ed il corrispondente diagramma temporale

I pin di cui parleremo da ora in poi sono, invece, quelli che servono per gestire eventuali periferici, direttamente od indirettamente, attraverso uno degli appositi chip a cui abbiamo fatto riferimento.

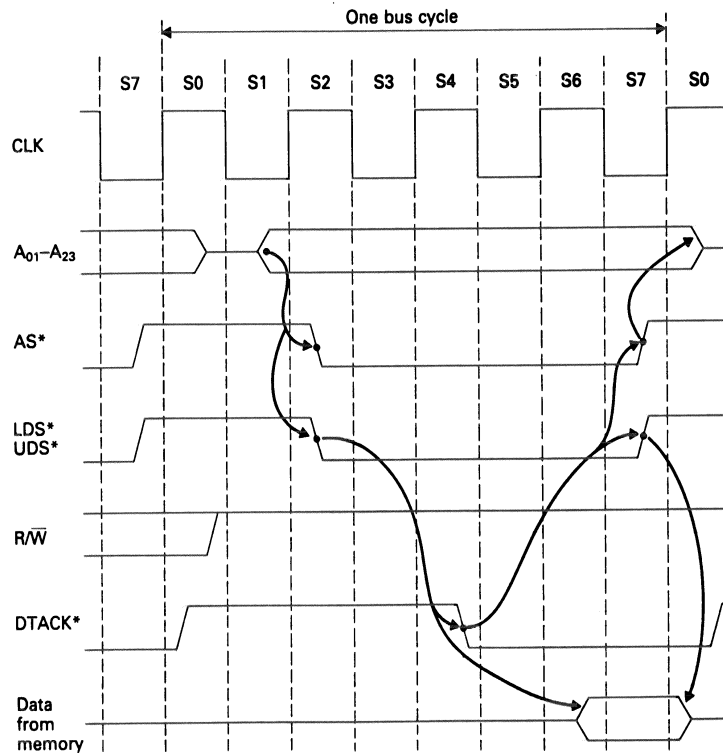


Fig. 11 - Diagramma temporale di un ciclo di read

BERR*: E' un ingresso che consente ad un dispositivo esterno, ad esempio la memoria, di informare la CPU che una qualche operazione non si è conclusa secondo le previsioni, mettendo bassa questa linea.

Si tratta di una possibilità che costituisce uno dei salti di qualità del 68.000 rispetto ai microprocessori della precedente generazione. Grazie a questo ingresso si possono facilmente superare situazioni, anche molto frequenti che, altrimenti, sarebbero catastrofiche. Basta pensare, ad esempio, ad un accesso ad una locazione di memoria difettosa o semplicemente inesistente. Quest'ultimo caso può verificarsi per errore dell'operatore o fortuitamente, per un segnale spurio che ha alterato il livello logico di una linea di indirizzo durante il trasferimento. Che cosa fa la CPU quando riceve questo segnale, non è banale dirlo, perchè dipende dal suo stato e dalle scelte che il sistema operativo è abilitato ad effettuare nei vari casi. Può, talvolta, semplicemente ritentare l'operazione.

C'è un gruppo di segnali esplicitamente dedicato al controllo dell'arbitrazione del bus.

Sappiamo già che cosa si intende per arbitraggio. E' quella funzione che diventa indispensabile quando nel sistema esiste almeno un altro dispositivo capace di gestire il protocollo del bus.

Non è necessario pensare a situazioni circuitali complicate, come sistemi multiprocessor, basta che nel sistema, basato sulla CPU 68.000, sia previsto anche semplicemente un "Direct Memory Access (DMA) controller che, quindi, in certi momenti deve poter gestire direttamente la memoria, mentre i corrispondenti piedini del processore sono floating.

BR*: E' il pin in entrata alla CPU, attraverso il quale l'altro potenziale master del bus comunica di aspirare a diventare "commander". E' quasi inutile dire che la linea deve essere gestita dai periferici (se più di uno) come linea "omnibus" (in logica negata con un resistore di pull-up e driver O.C. od O.D.).

BG*: E' il segnale che il 68.000, che sta svolgendo la funzione di commander, asserisce basso, per annunciare che sta per rilasciare il bus. Quando il master richiedente lo riceve, può eliminare il segnale di BR*. Inutile dire che nel frattempo una opportuna logica deve aver risolto eventuali conflitti tra master richiedenti.

BGACK*: E' il segnale che il master che ha ricevuto il "grant" invia in risposta al vecchio commander, per confermare di aver assunto il controllo del bus.

Altri pins interessanti sono quelli del codice di funzione, **FC0, FC1, FC2**. Le 3 linee in uscita che ne derivano possono essere considerate una sorta di estensione del bus indirizzi.

Esse in pratica manifestano all'esterno, con una opportuna temporizzazione, il fatto che la CPU sta eseguendo alcune specifiche operazioni che sono di interesse di uno o più periferici. Come si può vedere dalla tabella 2 la memoria è la principale beneficiaria delle linee FC.

FC2	FC1	FC0	Ciclo CPU
0	0	0	non definito
0	0	1	dati in modalità "user"
0	1	0	programma "user"
0	1	1	non definito
1	0	0	non definito
1	0	1	dati in modalità "supervisor"
1	1	0	programma "supervisor"
1	1	1	servizio interruzione

Tab. 2 - Codici di funzione

Dalla stessa tabella si vede che non tutte le combinazioni sono usate nel 68.000.

A parte il codice 111 (7₈), che serve ad informare i periferici che la CPU sta servendo un "*program interrupt*", gli altri 4 codici usati servono ad attivare la logica di verifica che l'accesso in memoria avvenga in un'area permessa. Il sistema 68.000, cioè, può avere due modalità di funzionamento, una riservata al "*supervisor*", che è l'unico che può accedere all' area riservata per i dati ed a quella per i programmi (sistema operativo), mentre l'utente comune ("*user*") può accedere solo alle aree non protette. Si badi che la possibilità di definire aree programmi ed aree

dati separate, in entrambe le modalità, contribuisce ad accrescere l'affidabilità del sistema, consentendo di organizzare un semplice meccanismo di protezione della memoria con segnalazione di errore in caso di sconfinamenti. Il 68.000 è stato il primo microprocessore single chip ad offrire questa possibilità, che fino all'apparizione di questa CPU, era stata riservata ai "*mainframe computers*" (macchine da grandi centri di calcolo).

Vedremo in altra sede che vi sono set di istruzioni assembler separati per le due modalità "*user*" e "*supervisor*".

Tre ingressi di controllo delle richieste di interrupt sono a disposizione dei periferici: **IPL0***, **IPL1***, **IPL2***. Poiché esse sono decodificate dalla CPU globalmente i livelli di priorità sono otto da 0 a 7. Quello 0 è il più basso, il 7 è il più alto. Tre bit di uno speciale registro della CPU, detto registro di stato (status register), fissano la soglia dinamica di livello di priorità di accettazione. Se il livello inviato dal dispositivo richiedente supera il livello di soglia, la sua richiesta può essere accettata. L'operazione di "settare" i bit di soglia nello status register si dice "mascheratura" dei periferici e può essere fatta, come già detto, dinamicamente.

Per completare il sommario esame dei piedini dell'MC 68.000 rimane da dire qualcosa sul gruppo di linee detto di "*controllo del bus sincrono*".

Questi segnali sono stati predisposti per consentire alla CPU 68.000, che ha una struttura dei controlli totalmente asincrona, di essere utilizzata con i chip di interfacciamento progettati per l'MC 6800 che, invece, aveva un bus sincrono.

VPA*: E' il segnale attivo basso che un periferico sincrono deve inviare alla CPU per comunicare che il proprio indirizzo sulle apposite linee è valido ed il trasferimento può iniziare. Quando la CPU riceve questo segnale avvia il trasferimento usando i pin VMA* ed E.

VMA*: E' il segnale di risposta della CPU, comunica al periferico che partecipa alla transazione che sulle linee A si trova un indirizzo valido.

E: E' un segnale di abilitazione e, soprattutto, di temporizzazione al quale il periferico si deve sincronizzare durante gli scambi. Il ciclo di E dura dieci cicli di clock del 68.000 ed è asimmetrico, in quanto è basso per sei cicli ed alto per quattro. Il clock del bus sincrono oscilla liberamente e continuamente rispetto ai cicli macchina della CPU. Il sincronismo si crea solo quando occorre, attraverso gli altri due segnali.