PD-32
Tecniche di Input/Output

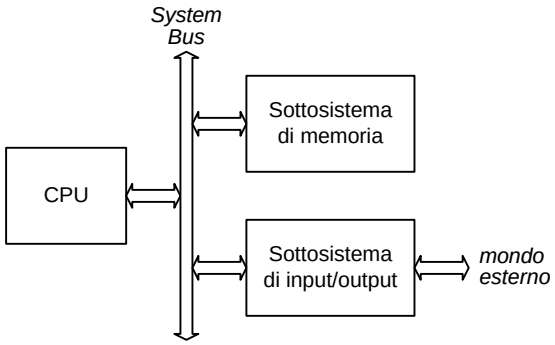


Fig. 1 Organizzazione di un generico sistema di elaborazione. L'unità centrale di elaborazione (*Central Processing Unit*, CPU) comunica col sottosistema di memoria e col sottosistema di Input/Output (I/O) attraverso il *bus di sistema* (*System Bus*); il sottosistema di Input/Output, a sua volta, comunica col mondo esterno.

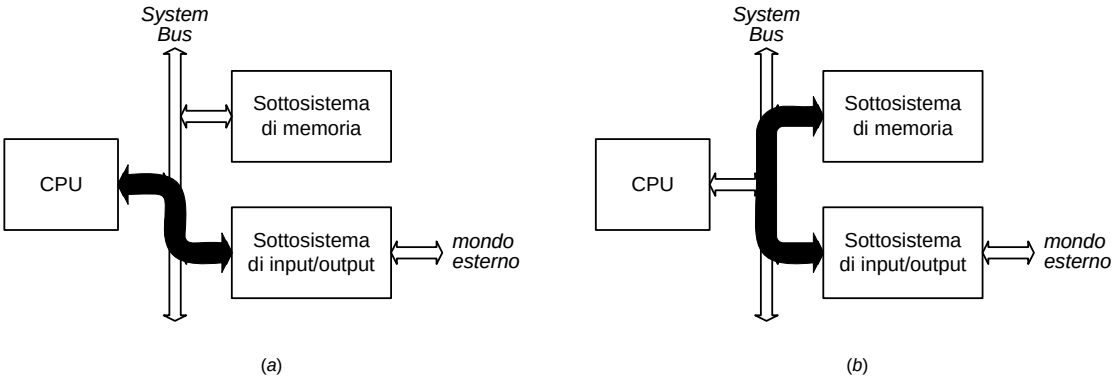


Fig. 2 Il trasferimento di dati tra il sistema di elaborazione e il mondo esterno può aver luogo lungo due possibili percorsi: (*a*) tra CPU e sottosistema di I/O, oppure (*b*) tra sottosistema di I/O e sottosistema di memoria, senza intervento della CPU. In quest'ultimo caso si parlerà di *accesso diretto alla memoria* (*Direct Memory Access*, DMA).

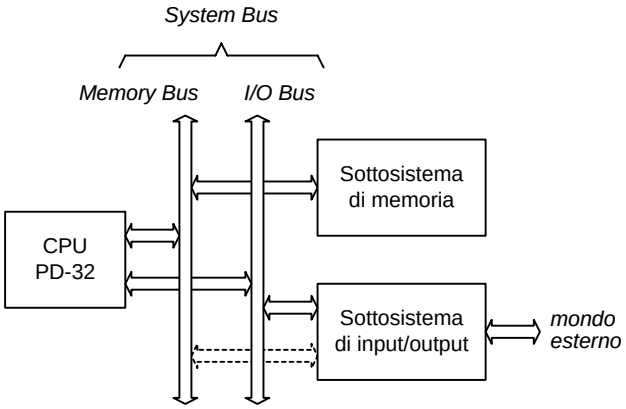


Fig. 3 Il System Bus del PD-32 è partizionato in un *Memory Bus*, che pone in comunicazione la CPU e il sottosistema di memoria, e da un *I/O Bus*, che pone in comunicazione la CPU col sottosistema di Input/Output. Se il sottosistema di I/O deve utilizzare la tecnica di accesso diretto alla memoria, esso dovrà potersi interfacciare anche al Memory Bus (connessione tratteggiata in figura).

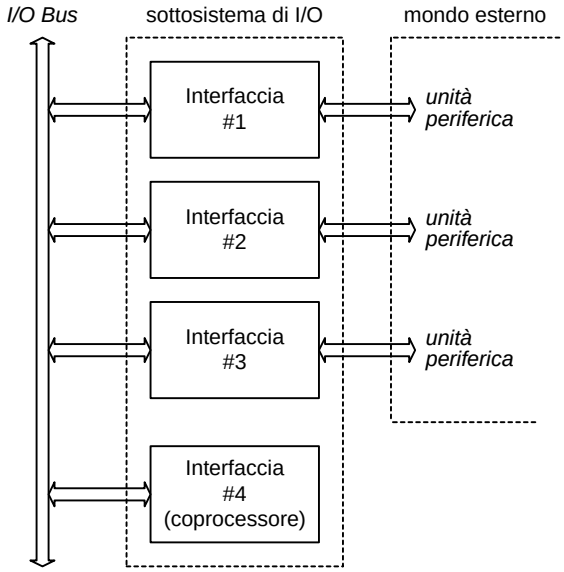


Fig. 4 Il sottosistema di I/O è costituito da una o più *interfacce*, tramite ciascuna delle quali una *unità periferica* (stampante, modem, etc.) può essere messa in comunicazione con l'I/O Bus. È anche possibile che un'interfaccia non comunichi con alcuna unità periferica, ma abbia lo scopo di ricevere dati dall'I/O Bus, elaborarli e poi restituire i risultati ancora attraverso l'I/O Bus; in questo caso l'interfaccia sarà genericamente (ma impropriamente) chiamata *coprocessore*.

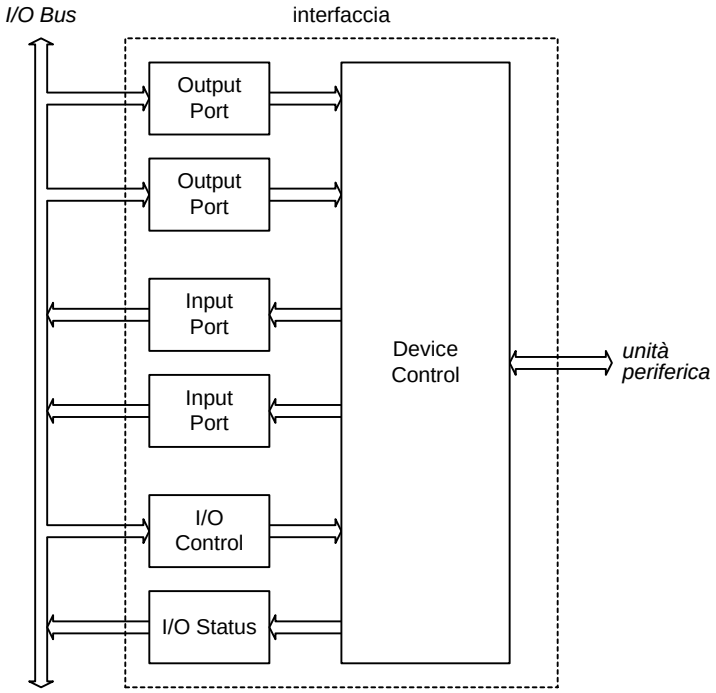


Fig. 5 Una generica interfaccia è costituita da:

- una o più *porte di uscita* (*output port*), dedicate a ricevere ed eventualmente immagazzinare un singolo dato (byte, word o longword) proveniente dall'I/O Bus;
- una o più *porte di ingresso* (*input port*), dedicate a trasmettere un singolo dato (byte, word o longword) dopo averlo eventualmente immagazzinato;
- una unità di *controllo di ingresso/uscita* (*I/O Control*) dedicata a dare avvio, dietro appositi comandi da parte della CPU, al complesso di operazioni previste per l'interfaccia, a controllare specifiche modalità operative dell'unità periferica, e ad altre funzioni analoghe;
- una unità di *stato di ingresso/uscita* (*I/O Status*), dedicata a segnalare alla CPU la fine del complesso di operazioni previste per l'interfaccia, ad informare la CPU dello stato dell'unità periferica, e ad altre funzioni analoghe;
- una unità di *controllo del dispositivo periferico* (*Device Control*), avente struttura strettamente dipendente dal dispositivo e dalle operazioni che su di esso devono essere eseguite, che è direttamente connessa all'unità periferica e che provvede a trasmettere ad essa dati e comandi, nonché a ricevere dati e informazioni di stato.

(Naturalmente, in circostanze e configurazioni particolari, una specifica interfaccia potrà essere priva di uno o più degli elementi indicati sopra.)

Una porta di input o di output viene identificata da un *indirizzo* (*address*), che deve essere globalmente unico a livello di sistema; in altri termini, non possono esistere due porte di output né due porte di input con lo stesso indirizzo, perché ciò, come si comprenderà meglio in seguito, comporterebbe dei conflitti nel corso delle relative operazioni di input/output; tuttavia, una porta di input può avere lo stesso indirizzo di una porta di output, poiché i due tipi di porte presuppongono due diverse transazioni di bus che non possono aver luogo contemporaneamente.

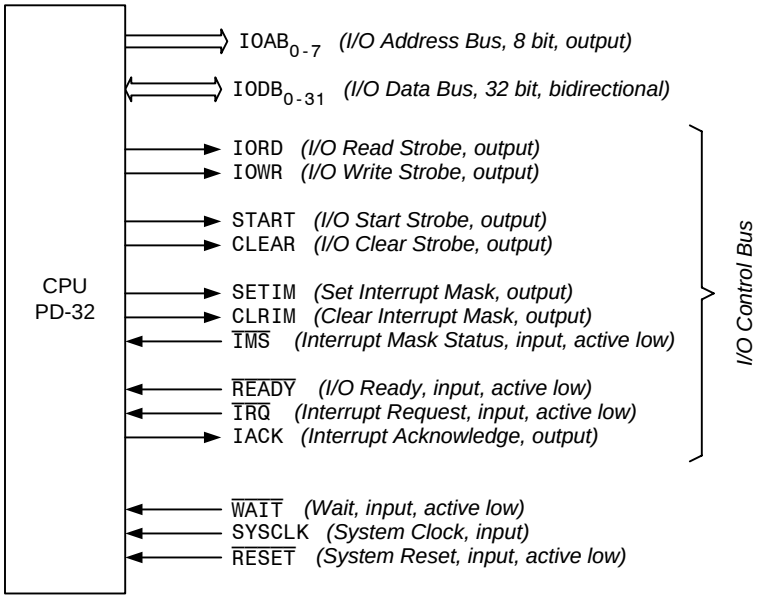


Fig. 6 Struttura dell'I/O Bus del PD-32:

- *I/O Address Bus* ($\texttt{IOAB}_{0\text{-}7}$) – gruppo di 8 segnali, generati dalla CPU, dedicati a trasmettere al sottosistema di I/O l'indirizzo della porta interessata alla transazione di bus;
- *I/O Data Bus* ($\texttt{IODB}_{0\text{-}31}$) – gruppo di 32 segnali, generati dalla CPU o dal sottosistema di I/O, dedicati a trasferire un dato (byte, word o longword) tra la CPU e una porta del sottosistema di I/O;
- *I/O Read Strobe* (IORD), *I/O Write Strobe* (IOWR) – generati dalla CPU, identificano una transazione di bus rispettivamente come *ciclo di I/O Input* oppure come *ciclo di I/O Output*;
- *I/O Start Strobe* (START), *I/O Clear Strobe* (CLEAR), *Set Interrupt Mask* (SETIM), *Clear Interrupt Mask* (CLRIM) – segnali generati dalla CPU nel corso di specifiche istruzioni PD-32;
- *I/O Ready* ($\overline{\texttt{READY}}$), Interrupt *Mask Status* ($\overline{\texttt{IMS}}$) – segnali che vengono campionati dalla CPU nel corso di specifiche istruzioni PD-32;
- *Interrupt Request* ($\overline{\texttt{IRQ}}$) – generato a cura del sottosistema di I/O, notifica alla CPU la presenza di almeno una richiesta di interruzione da parte di una o più interfacce;
- *Interrupt Acknowledge* (IACK) – generato dalla CPU nel momento in cui essa inizia a servire una richiesta di interruzione e deve identificare l'interfaccia che ha emesso la richiesta, per poter avviare una specifica *routine di servizio interruzione* (*Interrupt Service Routine*, ISR).

I segnali $\overline{\texttt{IORD}}$, $\overline{\texttt{IOWR}}$, $\overline{\texttt{START}}$, $\overline{\texttt{CLEAR}}$, $\overline{\texttt{SETIM}}$, $\overline{\texttt{CLRIM}}$, $\overline{\texttt{IMS}}$, $\overline{\texttt{READY}}$, $\overline{\texttt{IRQ}}$, $\overline{\texttt{IACK}}$ costituiscono nel loro complesso il *bus di controllo input/output* (*I/O Control Bus*)

I segnali:

- *Wait* ($\overline{\texttt{WAIT}}$) – generato a cura del sottosistema di I/O o del sottosistema di memoria nel corso di trasferimenti di dati che richiedono un tempo superiore al normale;
- *System Clock* (SYSCLK), *System Reset* ($\overline{\texttt{RESET}}$) – generati da appositi moduli speciali nel sistema di elaborazione;

possono eventualmente, se e quando necessario, essere utilizzati anche dal sottosistema di I/O.

```
INB dev_addr, dst        ; IODB[0-7] → dst[0-7]
INW dev_addr, dst        ; IODB[0-15] → dst[0-15]
INL dev_addr, dst        ; IODB[0-31] → dst[0-31]
```

Fig. 7 Istruzione `IN` per la lettura di un dato da una porta di input. Il campo sorgente `dev_addr` deve essere una costante, pari all'indirizzo della porta interessata; il campo destinazione `dst` può essere un registro di CPU oppure la specifica di una locazione di memoria, espressa in uno dei vari formati previsti dal set di istruzioni del PD-32. Gli opcode `INB`, `INW` o `INL` vengono utilizzati rispettivamente per la lettura di un byte (8 bit), una word (16 bit) o una longword (32 bit). Si noti come, nel trasferimento di un byte o di una word, vengano utilizzate solo le meno significative tra le 32 linee dell'I/O Data Bus.

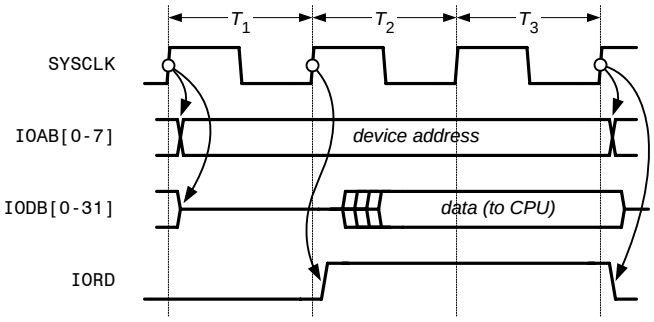


Fig. 8 Ciclo di I/O Read per la lettura di un dato da una porta di input. L'intero ciclo è sincronizzato con il System Clock, e dura esattamente 3 periodi di clock, indicati in figura con T_1, T_2 e T_3. All'inizio di T_1, la CPU emette sull'I/O Address Bus il *device address*, ossia l'indirizzo della porta di input interessata al trasferimento; nello stesso istante, la CPU dispone l'I/O Data Bus in modalità di ingresso, in modo da poter acquisire il dato quando questo verrà emesso dalla porta interessata. All'inizio di T_2 la CPU emette il segnale di I/O Read, qualificando così il ciclo come ciclo di lettura da input/output; in conseguenza, la porta di input identificata tramite l'indirizzo presente sull'I/O Address Bus emette sull'I/O Data Bus il dato richiesto, che deve mantenere valido e stabile per tutta la durata di `IORD`. Alla fine di T_3, la CPU acquisisce il dato presente sull'I/O Data Bus, cessa di emettere l'indirizzo di porta sull'I/O Address Bus e riporta nello stato inattivo lo strobe di I/O Read, concludendo così il ciclo. La durata del ciclo è commisurata alla possibilità, offerta anche a unità periferiche di velocità limitata, di predisporre con adeguati margini di tempo il dato da fornire alla CPU.

```
OUTB src, dev_addr          ; src[0-7] → IODB[0-7]
OUTW src, dev_addr          ; src[0-15] → IODB[0-15]
OUTL src, dev_addr          ; src[0-31] → IODB[0-31]
```

Fig. 9 Istruzione `OUT` per la scrittura di un dato su una porta di input. Il campo destinazione `dev_addr` deve essere una costante, pari all'indirizzo della porta interessata; il campo sorgente `src` può essere una costante, un registro di CPU oppure la specifica di una locazione di memoria, espressa in uno dei vari formati previsti dal set di istruzioni del PD-32. Gli opcode `OUTB`, `OUTW` o `OUTL` vengono utilizzati rispettivamente per la scrittura di un byte (8 bit), una word (16 bit) o una longword (32 bit). Si noti come, nel trasferimento di un byte o di una word, vengano utilizzate solo le meno significative tra le 32 linee dell'I/O Data Bus.

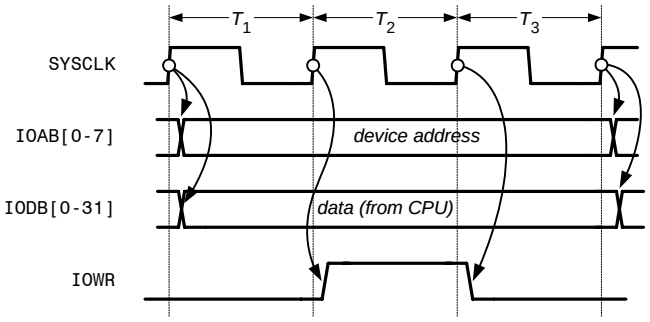


Fig. 10 Ciclo di I/O Write per la scrittura di un dato su una porta di output. L'intero ciclo è sincronizzato con il System Clock, e dura esattamente 3 periodi di clock, indicati in figura con T_1, T_2 e T_3. All'inizio di T_1, la CPU emette sull'I/O Address Bus il *device address*, ossia l'indirizzo della porta di input interessata al trasferimento; nello stesso istante, la CPU dispone l'I/O Data Bus in modalità di uscita e vi pone il dato da inviare alla porta di output interessata. All'inizio di T_2 la CPU emette il segnale di I/O Write, qualificando così il ciclo come ciclo di scrittura su input/output; in conseguenza, la porta di output identificata tramite l'indirizzo presente sull'I/O Address Bus preleva dall'I/O Data Bus il dato richiesto, che 1e mantenuto valido e stabile dalla CPU per tutta la durata di `IOWR`. Alla fine di T_2, la CPU riporta `IOWR` allo stato inattivo ma mantiene valido e stabile il dato sull'I/O Data Bus, per dar modo alla porta di output interessata di completare correttamente l'operazione di immagazzinamento. Alla fine di T_3, la CPU cessa di emettere l'indirizzo di porta sull'I/O Address Bus, concludendo così il ciclo. La durata del ciclo è commisurata alla possibilità, offerta anche a unità periferiche di velocità limitata, di disporre di ampi margini di tempo per catturare il dato fornito alla CPU.

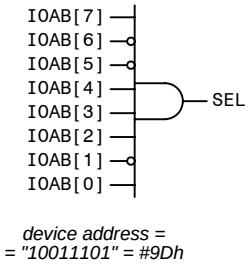


Fig. 11 Condizione necessaria affinché sia possibile l'accesso a una porta di input o di output, è che l'indirizzo assegnato alla porta coincida con quello che la CPU emette sull'I/O Address Bus nel corso di un ciclo di I/O Read o di I/O Write. In figura è illustrato un semplice decodificatore di indirizzo, basato su porta And, per verificare che sull'I/O Address Bus sia in transito l'indirizzo $10011101_2 = 9C_{16} = 156_{10}$; il segnale SEL, che viene distribuito all'interno dell'interfaccia, indica, quando attivo, che sull'I/O Address sta transitando l'indirizzo della porta interessata, che il ciclo di I/O in corso fa riferimento a quella porta, e funge pertanto da *abilitazione* per l'accesso ad essa.

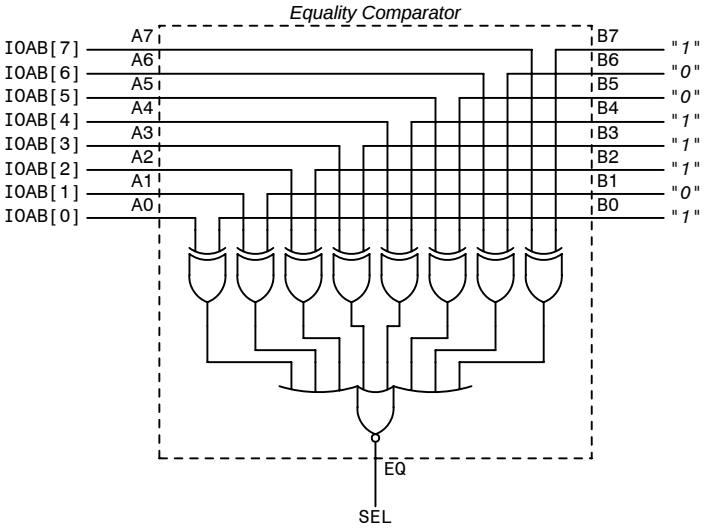


Fig. 12 Un altro metodo per la decodifica dell'indirizzo di porta fa utilizzo di un *comparatore di uguaglianza* (*Equality Comparator*) realizzato con otto XOR e un NOR; ciascun bit dell'I/O Address Bus viene confrontato con il corrispondente bit di una costante binaria, cablata all'interno dell'interfaccia; solo quando *tutti* i confronti danno esito positivo, il segnale SEL diventa attivo, abilitando in tal modo la porta all'accesso. Il vantaggio di questo metodo rispetto a quello illustrato in Fig. 11, al di là della maggior complessità circuitale, consiste in una maggiore flessibilità: lo stesso modulo può essere utilizzato su qualunque interfaccia, mentre il circuito di Fig. 11 va configurato in maniera specifica e distinta per ciascuna porta.

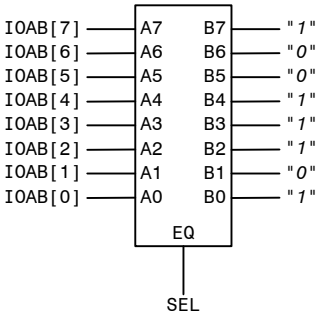
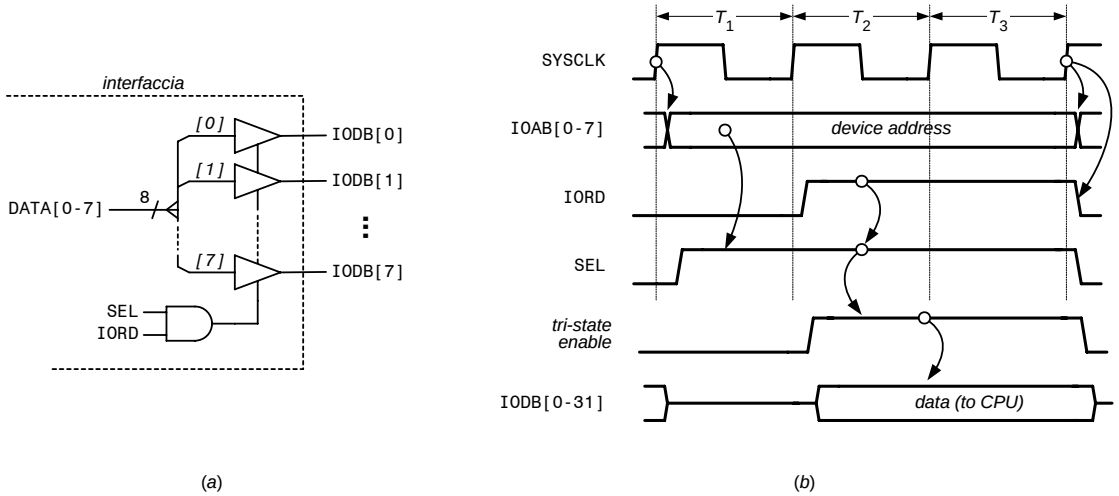
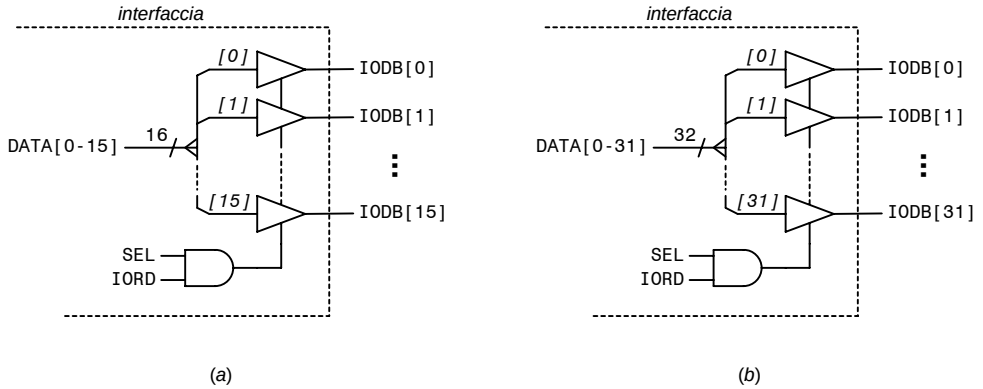


Fig. 13 Decodifica di indirizzo con comparatore di uguaglianza rappresentato come modulo.



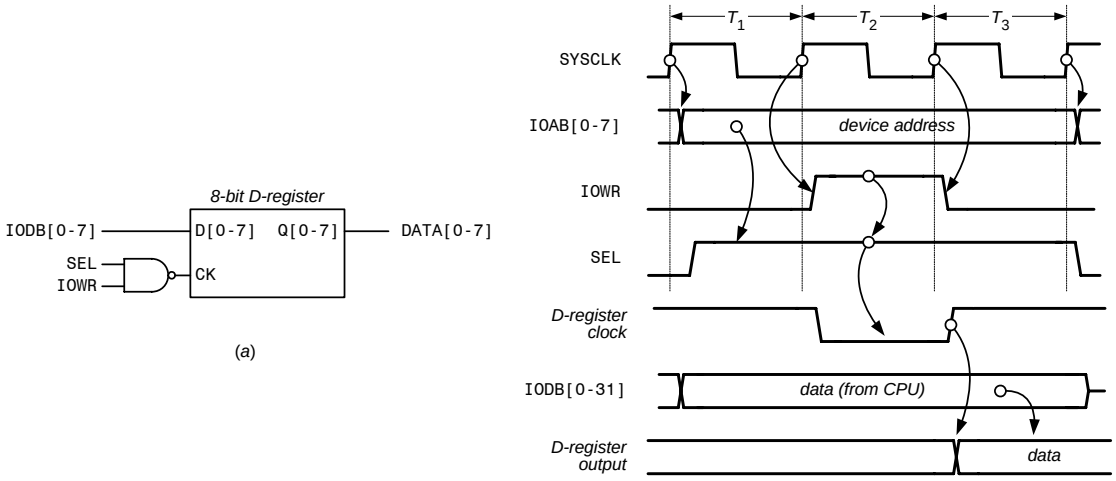
(*a*) (*b*)

Fig. 14 Struttura circuitale di una porta di input ad 8 bit (*a*) e relative temporizzazioni (*b*). Il byte `DATA[0-7]` generato all'interno dell'interfaccia (tipicamente, ma non necessariamente, prelevato dalle uscite di un registro) viene applicato a un array di buffer tri-state le cui uscite sono attestate sui bit 0-7 dell'I/O Data Bus; i buffer tri-state sono abilitati solo quando sull'I/O Address Bus è presente l'indirizzo della porta (e dunque `SEL` è attivo, cfr. Fig. 11 oppure Fig.) *e nello stesso tempo* è attivo `IORD`, con ciò indicando che è in corso un ciclo di I/O Read riferito a quella specifica porta di input. Si noti, sul diagramma di temporizzazione, come la porta emetta il dato esattamente quando richiesto dalla CPU nel corso del ciclo.



(*a*) (*b*)

Fig. 15 Porte di input (*a*) a 16 bit, (*b*) a 32 bit. La struttura è identica a quella di Fig. 14, salvo il numero di bit del dato inviato dall'interfaccia alla CPU.



(*a*)

(*b*)

Fig. 16 Struttura circuitale di una porta di output da 8 bit (*a*) realizzata con registro D edge-triggered, e relative temporizzazioni (*b*). Poiché il registro cattura i dati sul fronte di salita del clock, questo deve aver luogo in corrispondenza del fronte di caduta di `IOWR`, ossia all'ultimo momento utile per l'acquisizione del dato dall'I/O Data Bus; di conseguenza, il clock al registro viene generato mediante un NAND, pilotato dal segnale `SEL`, che è la decodifica dell'indirizzo presente sull'I/O Address Bus, e dal segnale `IOWR`, che qualifica il ciclo di I/O Write. Il dato verrà presentato alle uscite del registro `DATA[0-7]` e da qui distribuito all'interfaccia.

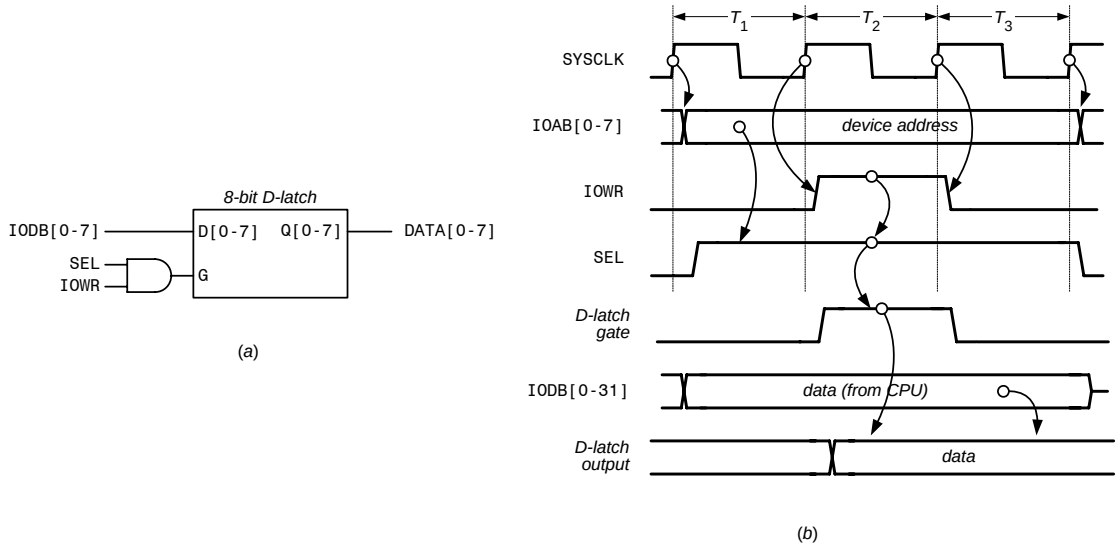


Fig. 17 Struttura circuitale di una porta di output da 8 bit (*a*) realizzata con latch D, e relative temporizzazioni (*b*). Poiché il registro cattura i dati presenti sull'I/O Data Bus quando il gate G è attivo, questo deve aver luogo in corrispondenza di `IOWR`; di conseguenza, il clock al registro viene generato mediante un AND, pilotato dal segnale `SEL`, che è la decodifica dell'indirizzo presente sull'I/O Address Bus, e dal segnale `IOWR`, che qualifica il ciclo di I/O Write. Il dato verrà presentato alle uscite del registro `DATA[0-7]` e da qui distribuito all'interfaccia. Non vi sono sostanziali differenze di comportamento tra questo circuito e quello di Fig. 16, salvo l'istante in cui il dato presente sull'I/O Data Bus viene trasferito sulle uscite `DATA[0-7]`.

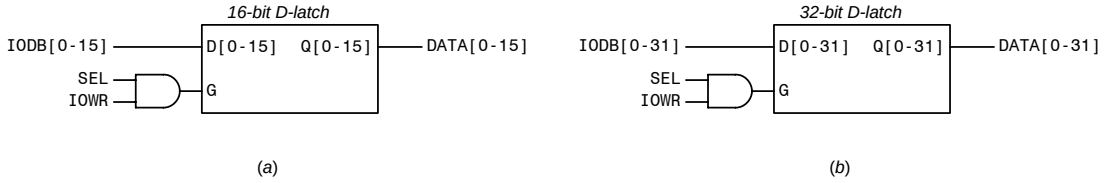


Fig. 18 Struttura circuitale di una porta di output da 16 bit (*a*) e da 32 bit (*b*). Non vi sono sostanziali differenze col circuito di Fig. 17, salvo la dimensione dei registri.

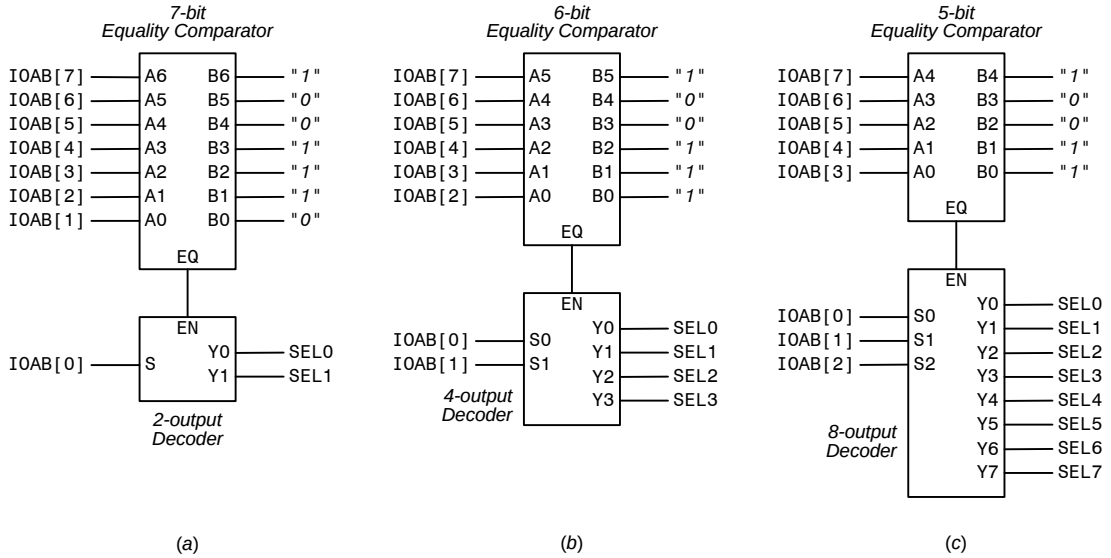


Fig. 19 Una singola interfaccia può contenere più di una porta di input, oppure più di una porta di output; sebbene ad una porta di input e ad una di output possa essere assegnato lo stesso indirizzo, due o più porte di output oppure due o più porte di input devono essere allocate a indirizzi diversi. In casi del genere si può assumere che gli indirizzi assegnati alle porte siano consecutivi, in modo che gli 8 bit di indirizzo che transita sull'I/O Address Bus possano essere suddivisi in una parte che rimane fissa per tutti gli indirizzi delle porte residenti sull'interfaccia, mentre la restante parte è variabile da una porta all'altra. Di conseguenza, i vari `SEL` necessari possono essere generati decodificando la parte fissa mediante un comparatore di uguaglianza, la cui uscita viene applicata all'abilitazione di un decoder, il quale riceve in ingresso la parte variabile dell'indirizzo e produce di conseguenza in uscita i `SEL` desiderati. Questa tecnica è illustrata in (*a*) per generare i due `SEL` corrispondenti agli indirizzi $10011100_2 = 9C_{16}$ (`SEL0`) e $10011101_2 = 9D_{16}$ (`SEL1`); in (*b*) per generare i quattro `SEL` corrispondenti agli indirizzi $10011100_2 = 9C_{16}$, $10011101_2 = 9D_{16}$, $10011110_2 = 9E_{16}$, $10011111_2 = 9F_{16}$; in (*c*) per generare gli otto `SEL` corrispondenti agli indirizzi da $10011000_2 = 98_{16}$ a $10011111_2 = 9F_{16}$.

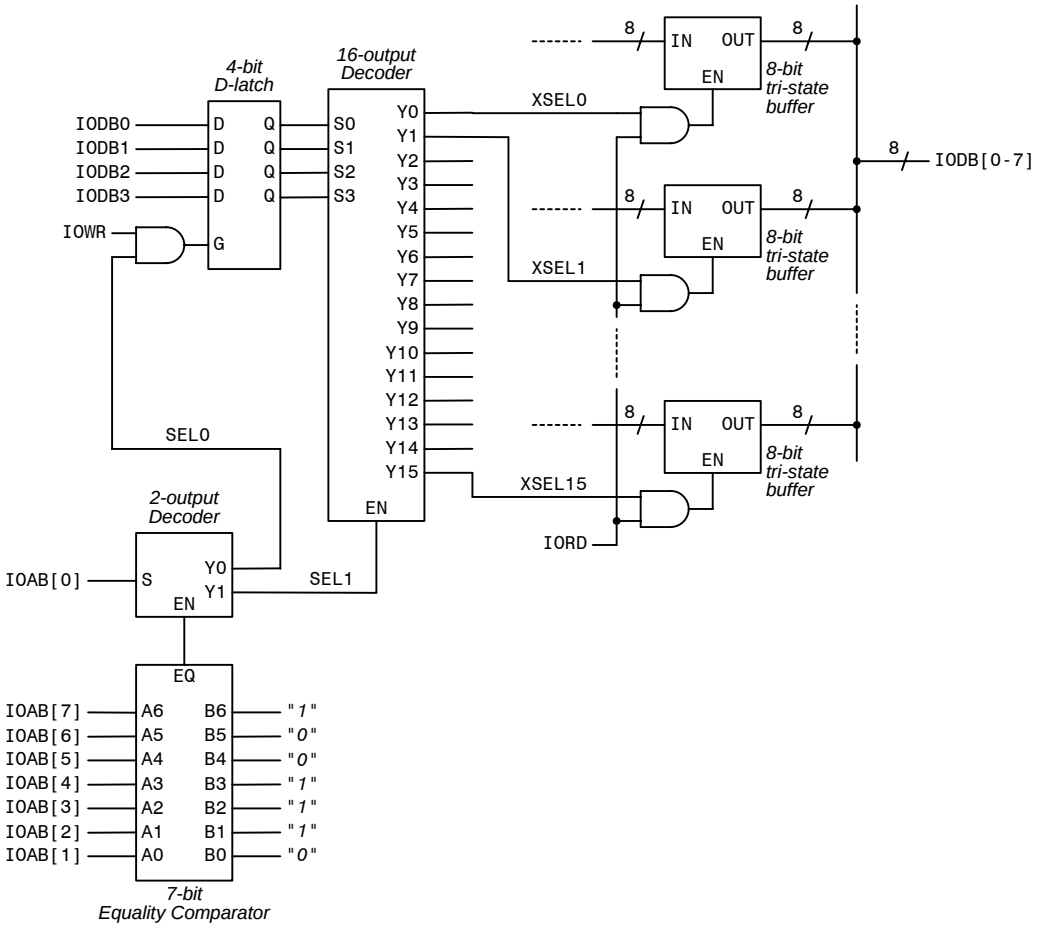


Fig. 20 Lo spazio degli indirizzi definiti attraverso l'I/O Address Bus è limitato a 256 "locazioni"; di conseguenza, è opportuno che una singola interfaccia impegni il minor numero possibile di indirizzi di I/O; per di più, è possibile che una singola interfaccia debba disporre di un numero di porte (di input e/o di output) anche superiore a 256. In circostanze del genere, per limitare il numero di indirizzi di I/O impegnati da una singola interfaccia, si ricorre alla tecnica del *multiplexing delle porte*, che consiste nel predisporre una porta di output avente le funzioni di selettore per una tra le *N* porte disponibili sull'interfaccia. Si consideri l'esempio riportato in figura: se la CPU vuole accedere alla porta di input n. 5 tra le 16 disponibili, essa scriverà prima il numero 5 (indice della porta) all'indirizzo $10011100_2 = 9C_{16}$ al quale è allocato il D-latch che funge da registro selettore, mediante l'istruzione

```
outb #5, #9Ch  ; scrittura dell'indice
```

In tal modo, il valore dell'indice viene applicato al decoder a 16 uscite; quando successivamente la CPU richiede una lettura dalla porta di indirizzo $10011101_2 = 9D_{16}$

```
inb #9Dh, r1  ; porta selezionata -> R1
```

il decoder a 16 ingressi viene abilitato tramite `SEL1` e, con esso, viene anche abilitata la porta di input n. 5 controllata dall'uscita di decoder `XSEL5`; in tal modo il dato presente all'ingresso del buffer tri-state n. 5 viene avviato verso l'I/O Data Bus. L'operazione di input viene di conseguenza suddivisa in un'operazione di output per l'emissione dell'indice, e in un'operazione di input propriamente detta.

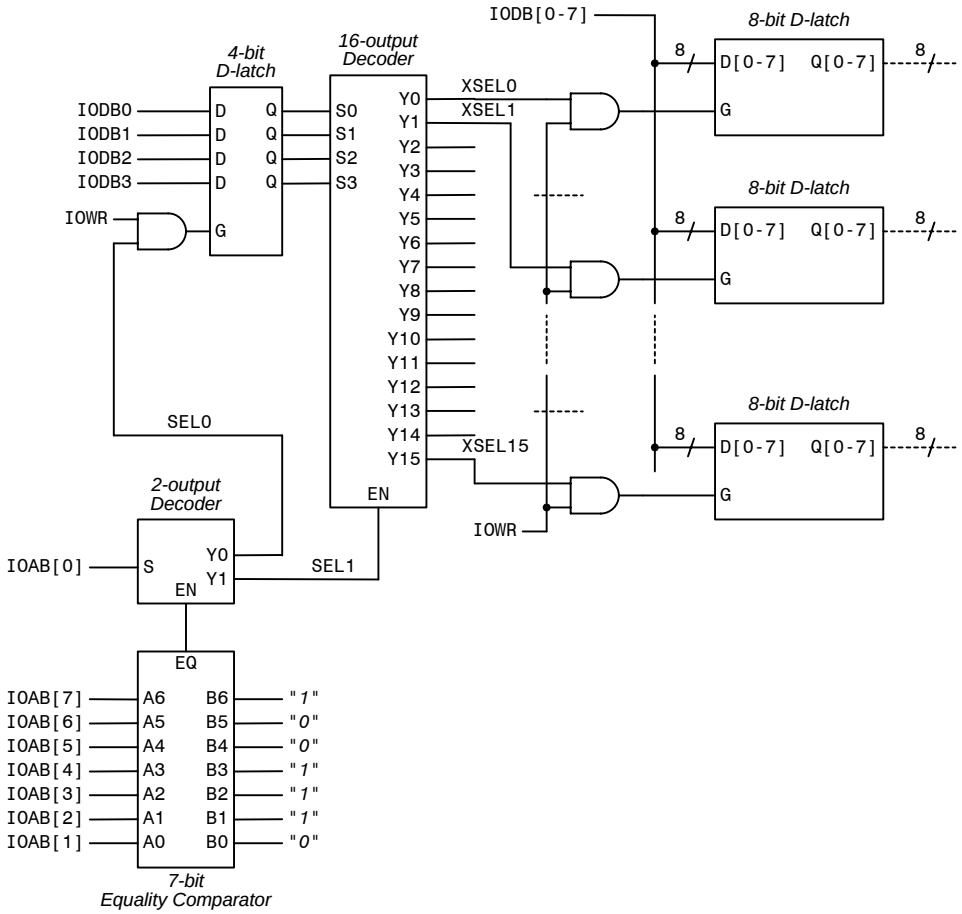


Fig. 21 La tecnica del multiplexing delle porte applicata a 16 porte di output a 8 bit. Il circuito non differisce molto da quello di Fig. 20, salvo che i segnali interni `XSEL` diventano abilitazione per la scrittura su porte di output anziché abilitazione alla lettura di porte di input. Volendo accedere alla porta di indice 5, la CPU eseguirà la seguente sequenza di operazioni:

```
outb #5, #9Ch  ; scrittura dell'indice
outb r1, #9Dh  ; R1 -> porta selezionata
```

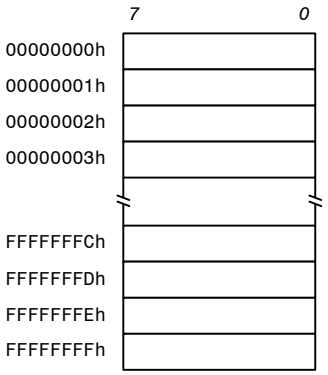


Fig. 22 La memoria del PD-32 è organizzata logicamente come un array di $2^{32} = 4\,294\,967\,296$ *parole* da 8 bit (*byte*), ciascuna identificata univocamente da un numero (*indirizzo*, *address*) espresso su 32 bit; nella figura gli indirizzi sono rappresentati in esadecimale. Due parole sono considerate consecutive se i loro indirizzi sono consecutivi.

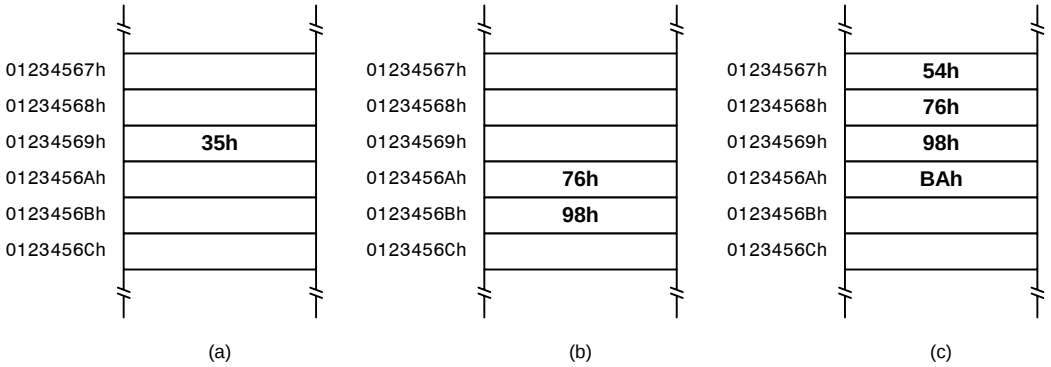


Fig. 23 I dati manipolati dal PD-32 possono essere costituiti da 8 bit (*byte*), da 16 bit (*word*) oppure da 32 bit (*longword*). Nel caso di dati *multibyte* (word oppure longword), il dato viene suddiviso in byte che vengono allocati in memoria entro parole consecutive, a partire dal byte meno significativo e, via via agli indirizzi successivi, a finire col byte più significativo; l'indirizzo del dato coincide con l'indirizzo al quale è allocato il suo byte meno significativo (*convenzione little-endian*). Ad esempio: (*a*) il byte `35h` è allocato all'indirizzo `01234569h`; (*b*) la word `9876h` è allocata all'indirizzo `0123456Ah`: essa viene suddivisa in due byte `76h` e `98h` che vengono allocati rispettivamente agli indirizzi `0123456Ah` e `0123456Bh`; (*c*) la longword `BA987654h` è allocata all'indirizzo `01234567h`: essa viene suddivisa nei quattro byte `54h`, `76h`, `98h` e `BAh` che vengono allocati rispettivamente agli indirizzi consecutivi `01234567h`, `01234568h`, `01234569h` e `0123456Ah`, a partire dal byte meno significativo `54h` e a finire col byte più significativo `BAh`.

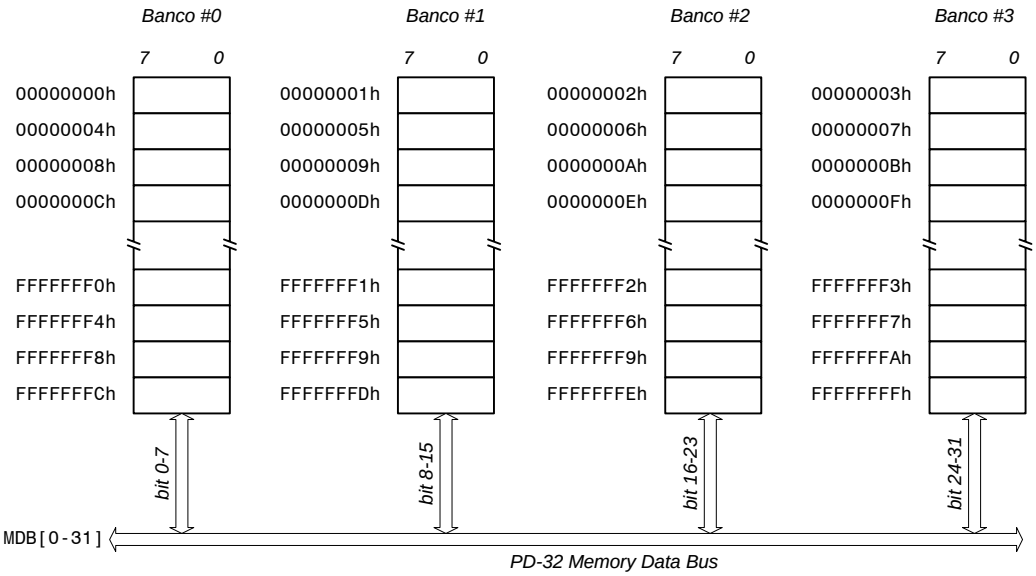


Fig. 24 L'organizzazione fisica della memoria è diversa dall'organizzazione logica illustrata in Fig. 22: poiché il Memory Data Bus del PD-32 ($\texttt{MDB}_{0-31}$) è a 32 bit, la memoria viene fisicamente organizzata come una matrice di $2^{30} \times 4$ byte, dove ciascuna colonna (che prende il nome di *banco di memoria*) interagisce con una specifica porzione del Memory Data Bus; in particolare:

- la colonna costituita dalle parole il cui indirizzo $A \equiv 0 \pmod 4$ (*A* è *congruo a 0 modulo 4*, ossia il resto della divisione di *A* per 4 è pari a 0), detta *Banco #0*, interagisce con i bit 0-7 del Memory Data Bus;
- la colonna costituita dalle parole il cui indirizzo $A \equiv 1 \pmod 4$ (*A* è *congruo a 1 modulo 4*, ossia il resto della divisione di *A* per 4 è pari a 1), detta *Banco #1*, interagisce con i bit 8-15 del Memory Data Bus;
- la colonna costituita dalle parole il cui indirizzo $A \equiv 2 \pmod 4$, detta *Banco #2*, interagisce con i bit 16-23 del Memory Data Bus;
- la colonna costituita dalle parole il cui indirizzo $A \equiv 3 \pmod 4$, detta *Banco #3*, interagisce con i bit 24-31 del Memory Data Bus.

L'accesso alla memoria in un singolo cic*lo (ciclo di memoria*, *Memory Cycle*) viene allora realizzato *selezionando un'intera riga* della matrice e *abilitando uno o più banchi* a comunicare col Memory Data Bus.

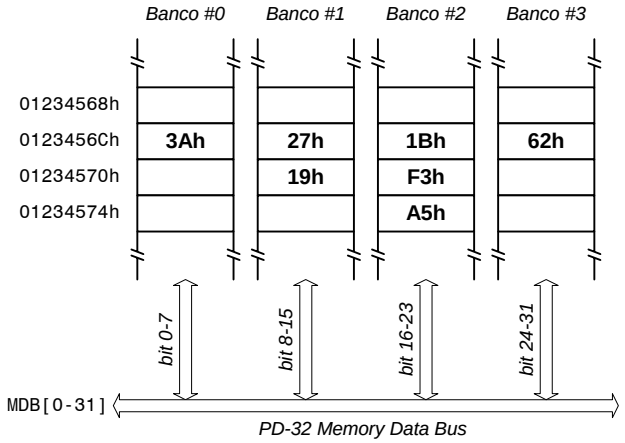


Fig. 25 Esempi di accesso alla memoria in un singolo ciclo:

- Per accedere alla longword `621B273Ah` allocata all'indirizzo `0123456Ch`, viene selezionata la riga della matrice di memoria di indirizzo iniziale `0123456Ch` e tutti e quattro i banchi vengono abilitati: in tal modo i singoli byte della longword vengono inviati al Memory Data Bus nella corretta disposizione.
- Per accedere alla word `F319h` allocata all'indirizzo `012345671h`, viene selezionata la riga di indirizzo iniziale `012345670h` e vengono abilitati solo i banchi 1 e 2; la word si dispone così sui bit 8-23 del Memory Data Bus, e sarà compito dei circuiti interni alla CPU riposizionarli correttamente sui bit 0-15 prima di avviarli a destinazione.
- Per accedere al byte `A5h` allocato all'indirizzo `012345676h`, viene selezionata la riga di indirizzo `012345674h` e viene abilitato il solo banco 2; in tal modo, il byte si dispone sui bit 16-23 del Memory Data Bus, e sarà poi compito dei circuiti interni della CPU riposizionarlo correttamente sui bit 0-7 prima di avviarlo a destinazione.

Nel caso di accesso in scrittura, anziché in lettura, i circuiti interni alla CPU dovranno in primo luogo posizionare correttamente il dato (byte, word o longword) sul Memory Data Bus in funzione dell'indirizzo di memoria al quale esso dovrà essere scritto, dopo di che la selezione della riga e l'abilitazione dei banchi procederanno in maniera analoga al caso della lettura.

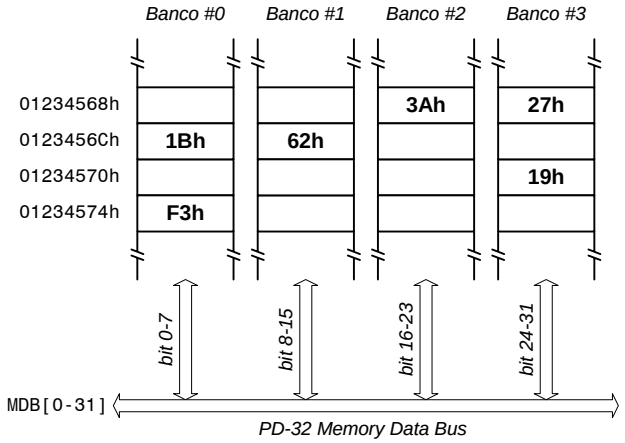


Fig. 26 È possibile che, a seconda di come i dati sono allocati in memoria, la CPU non possa accedere ad essi in un singolo ciclo, ma debba risolvere l'accesso in due cicli successivi, poiché il dato interessato è ripartito su righe diverse della matrice. Ad esempio:

- Per accedere alla longword `621B273Ah` allocata all'indirizzo `0123456Ah`:
  1. in un primo ciclo viene selezionata la riga della matrice di memoria di indirizzo iniziale `01234568h` e vengono abilitati solo i banchi 2 e 3: in tal modo i bit 0-15 della longword si dispongono sui bit 16-31 del Memory Data Bus;
  2. in un secondo ciclo successivo viene selezionata la riga della matrice di memoria di indirizzo iniziale `0123456Ch` e vengono abilitati solo i banchi 0 e 1: in tal modo i bit 16-31 della longword si dispongono sui bit 0-15 del Memory Data Bus.
- Per accedere alla word `F319h` allocata all'indirizzo `012345673h`:
  1. in un primo ciclo viene selezionata la riga di indirizzo iniziale `012345670h` e viene abilitato solo il banco 3; i bit 0-7 della word si dispongono così sui bit 24-31 del Memory Data Bus;
  2. in un secondo ciclo successivo viene selezionata la riga di indirizzo iniziale `012345674h` e viene abilitato solo il banco 0; i bit 8-15 della word si dispongono così sui bit 0-7 del Memory Data Bus.

In tutti i casi, sarà compito della CPU sia memorizzare in un registro temporaneo il dato parziale acquisito nel primo ciclo, sia redisporre i dati ottenuti nei due cicli in posizione corretta prima di avviarli a destinazione. (È appena il caso di osservare che il doppio ciclo di accesso *può* essere richiesto solo per dati da 16 o 32 bit, *mai* per dati da 8 bit.)

Nel caso di accesso in scrittura, anziché in lettura, i circuiti interni alla CPU dovranno in primo luogo separare il dato nelle due componenti che andranno allocate su righe della matrice di memoria, poi generare due distinti cicli di memoria per scrivere separatamente le due componenti, selezionando ogni volta una riga di memoria e abilitando i banchi interessati.

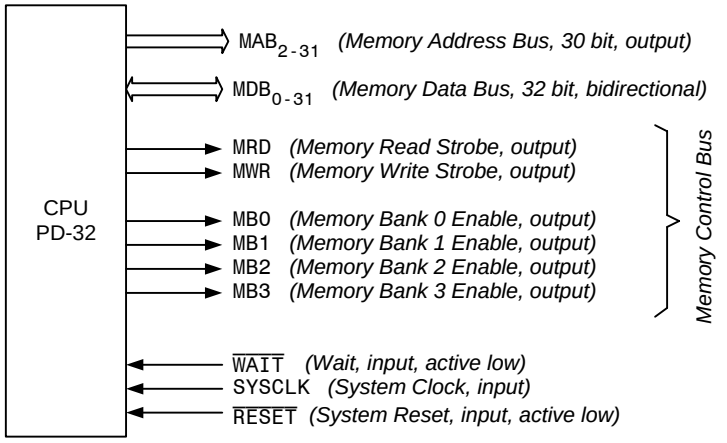


Fig. 27 Struttura del Memory Bus del PD-32:

- *Memory Address Bus* (MAB_{2-31}) – gruppo di 30 segnali, generati dalla CPU, dedicati a trasmettere al sottosistema di memoria l'indirizzo della riga della matrice di memoria interessata alla transazione di bus (si noti come, essendo gli indirizzi di riga sempre esattamente divisibili per 4, e avendo di conseguenza i bit 0-1 sempre uguali a zero, tali bit *non vengano emessi*);
- *Memory Data Bus* (MDB_{0-31}) – gruppo di 32 segnali, generati dalla CPU, dedicati a trasferire un dato (byte, word o longword) tra la CPU e il sottosistema di memoria;
- *Memory Read Strobe* (MRD) – generato dalla CPU, identifica una transazione di bus come *ciclo di Memory Read*;
- *Memory Write Strobe* (MWR) – generato dalla CPU, identifica una transazione di bus come *ciclo di Memory Write*;
- *Memory Bank Enable* (MB0, MB1, MB2, MB3) – segnali generati dalla CPU per abilitare uno o più banchi del sottosistema di memoria.

I segnali MRD, MWR, MB0, MB1, MB2, MB3 costituiscono nel loro complesso il *bus di controllo memoria* (*Memory Control Bus*)

I segnali:

- *Wait* ($\overline{\text{WAIT}}$) – generato a cura del sottosistema di I/O o del sottosistema di memoria nel corso di trasferimenti di dati che richiedono un tempo superiore al normale;
- *System Clock* (SYSCLK), *System Reset* ($\overline{\text{RESET}}$) – generati da appositi moduli speciali nel sistema di elaborazione;

possono eventualmente, se e quando necessario, essere utilizzati anche dal sottosistema di memoria.

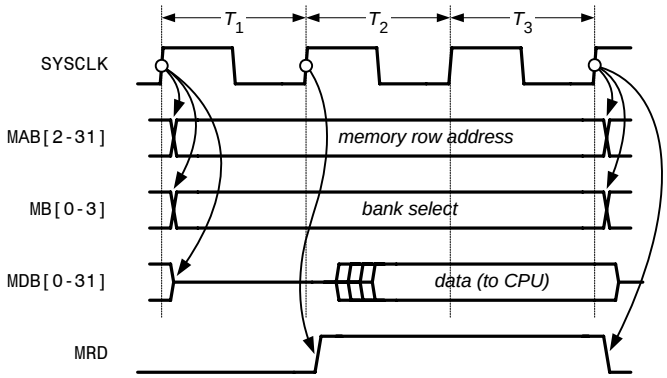


Fig. 28 Ciclo di *Memory Read*. Come il ciclo di I/O Read (Fig. 8), esso ha durata pari a 3 cicli di System Clock, indicati con T_1, T_2 e T_3. All'inizio di T_1, la CPU applica sul Memory Address Bus ($\texttt{MAB}_{2\text{-}31}$) i bit dell'indirizzo della locazione interessata (con esclusione dei due bit meno significativi) e, a seconda del tipo di dato (byte, word o longword) e della sua disposizione all'interno della riga di memoria (quest'ultima informazione viene ricavata dall'analisi dei due bit meno significativi dell'indirizzo), comanda l'abilitazione dei banchi di memoria interessati applicando un'opportuna configurazione su $\texttt{MB}_{0\text{-}3}$; inoltre dispone il Memory Data Bus ($\texttt{MDB}_{0\text{-}31}$) a ricevere dati dal sottosistema di memoria. All'inizio di T_2, la CPU attiva il Memory Read Strobe MRD, in conseguenza del quale il sottosistema di memoria presenta sul Memory Data Bus le informazioni estratte dalle locazioni selezionate. Al termine di T_3, la CPU cattura i dati trovati sul Memory Data Bus e riporta il Memory Read Strobe inattivo. Si noti come per tutta la durata del ciclo il Memory Address Bus e i segnali di Memory Bank Select rimangano validi e stabili.

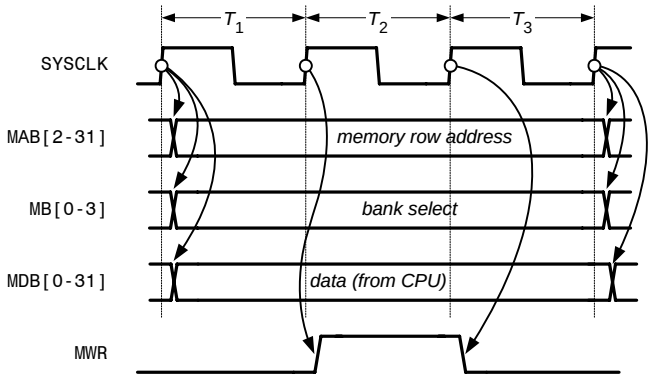


Fig. 29 Ciclo di *Memory Write*. Come il ciclo di I/O Write (Fig. 10), esso ha durata pari a 3 cicli di System Clock, indicati con T_1, T_2 e T_3. All'inizio di T_1, la CPU applica sul Memory Address Bus ($\texttt{MAB}_{2\text{-}31}$) i bit dell'indirizzo della locazione interessata (con esclusione dei due bit meno significativi) e, a seconda del tipo di dato (byte, word o longword) e della sua disposizione finale all'interno della riga di memoria (quest'ultima informazione viene ricavata dall'analisi dei due bit meno significativi dell'indirizzo), comanda l'abilitazione dei banchi di memoria interessati applicando un'opportuna configurazione su $\texttt{MB}_{0\text{-}3}$; inoltre emette sul Memory Data Bus ($\texttt{MDB}_{0\text{-}31}$) il dato che andrà scritto in memoria. All'inizio di T_2, la CPU attiva il Memory Write Strobe `MWR`, in conseguenza del quale il sottosistema di memoria avvia le operazioni di scrittura nelle locazioni selezionate. All'inizio di T_3, la CPU riporta inattivo lo strobe di scrittura, col che il sottosistema di memoria porta a completamento le operazioni di scrittura; per consentire sufficienti margini di tempo per le operazioni coinvolte, la CPU lascia ancora valide e stabili fino alla fine di T_3 sia le informazioni di indirizzo e di banco che il dato da scrivere in memoria.

Dato	***Indirizzo A***	***Operazioni di bus***
Byte (8 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-7} \rightarrow dst_{0-7}$
	$A \equiv 1 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0100 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{8-15} \rightarrow dst_{0-7}$
	$A \equiv 2 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0010 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{16-23} \rightarrow dst_{0-7}$
	$A \equiv 3 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{24-31} \rightarrow dst_{0-7}$
Word (16 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1100 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-15} \rightarrow dst_{0-15}$
	$A \equiv 1 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0110 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{8-23} \rightarrow dst_{0-15}$
	$A \equiv 2 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0011 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{16-31} \rightarrow dst_{0-15}$
	$A \equiv 3 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{24-31} \rightarrow dst_{0-7}$ $[A+4] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-7} \rightarrow dst_{8-15}$
Longword (32 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1111 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-31} \rightarrow dst_{0-31}$
	$A \equiv 1 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0111 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{8-31} \rightarrow dst_{0-23}$ $[A+4] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-7} \rightarrow dst_{24-31}$
	$A \equiv 2 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0011 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{16-31} \rightarrow dst_{0-15}$ $[A+4] \rightarrow MAB_{2-31},\ 1100 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-15} \rightarrow dst_{16-31}$
	$A \equiv 3 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{24-31} \rightarrow dst_{0-7}$ $[A+4] \rightarrow MAB_{2-31},\ 1110 \rightarrow MB_{0-3},\ MRD = 1,\ MDB_{0-23} \rightarrow dst_{8-31}$

Fig. 30 Come accennato in precedenza (Fig. 26), la disposizione di un dato in memoria può essere tale che un singolo ciclo di Memory Read non sia sufficiente ad accedere all'intero dato, poiché esso non è contenuto in una singola riga della matrice di memoria. In questa tavola sono riportati tutti i possibili casi che si possono verificare, con le rispettive operazioni di bus necessarie per leggere il dato in maniera completa; *A* rappresenta l'indirizzo del dato, [*A*] è pari al quoziente della divisione intera di *A* per 4 (ottenuto semplicemente da *A* eliminandone i due bit meno significativi di indice 0, 1) e `dst` rappresenta la destinazione all'interno della CPU dove il dato letto viene immagazzinato.

Un dato si dice *allineato alla longword* se il suo indirizzo è esattamente divisibile per 4, mentre si dice *allineato alla word* se il suo indirizzo è esattamente divisibile per 2. Si noti che un dato allineato alla longword è anche allineato alla word, mentre un dato allineato alla word può non essere anche allineato alla longword. È facile ricavare dalla tavola che:

- La lettura di un byte è sempre risolta in un singolo ciclo.
- La lettura di una word è sempre risolta in un singolo ciclo, a meno che il suo indirizzo, diviso per 4, non dia come resto 3.
- La lettura di una longword è risolta in un singolo ciclo solo se essa è allineata alla longword.

Dato	*Indirizzo A*	*Operazioni di bus*
Byte (8 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{0-7},\ MWR = 1$
	$A \equiv 1 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0100 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{8-15},\ MWR = 1$
	$A \equiv 2 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0010 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{16-23},\ MWR = 1$
	$A \equiv 3 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{24-31},\ MWR = 1$
Word (16 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1100 \rightarrow MB_{0-3},\ src_{0-15} \rightarrow MDB_{0-15},\ MWR = 1$
	$A \equiv 1 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0110 \rightarrow MB_{0-3},\ src_{0-15} \rightarrow MDB_{8-23},\ MWR = 1$
	$A \equiv 2 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 0011 \rightarrow MB_{0-3},\ src_{0-15} \rightarrow MDB_{16-31},\ MWR = 1$
	$A \equiv 3 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{24-31},\ MWR = 1$ $[A+4] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ src_{8-15} \rightarrow MDB_{0-7},\ MWR = 1$
Longword (32 bit)	$A \equiv 0 \pmod 4$	*Singolo ciclo:* $[A] \rightarrow MAB_{2-31},\ 1111 \rightarrow MB_{0-3},\ src_{0-31} \rightarrow MDB_{0-31},\ MWR = 1$
	$A \equiv 1 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0111 \rightarrow MB_{0-3},\ src_{0-23} \rightarrow MDB_{8-31},\ MWR = 1$ $[A+4] \rightarrow MAB_{2-31},\ 1000 \rightarrow MB_{0-3},\ src_{24-31} \rightarrow MDB_{0-7},\ MWR = 1$
	$A \equiv 2 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0011 \rightarrow MB_{0-3},\ src_{0-15} \rightarrow MDB_{16-31},\ MWR = 1$ $[A+4] \rightarrow MAB_{2-31},\ 1100 \rightarrow MB_{0-3},\ src_{16-31} \rightarrow MDB_{0-15},\ MWR = 1$
	$A \equiv 3 \pmod 4$	*Doppio ciclo:* $[A] \rightarrow MAB_{2-31},\ 0001 \rightarrow MB_{0-3},\ src_{0-7} \rightarrow MDB_{24-31},\ MWR = 1$ $[A+4] \rightarrow MAB_{2-31},\ 1110 \rightarrow MB_{0-3},\ src_{8-31} \rightarrow MDB_{0-23},\ MWR = 1$

Fig. 31 Operazioni di bus necessarie per la scrittura di un dato in memoria, in funzione della sua dimensione e dell'indirizzo al quale esso deve essere allocato. *A* rappresenta l'indirizzo del dato, [*A*] è pari al quoziente della divisione intera di *A* per 4 (ottenuto semplicemente da *A* eliminandone i due bit meno significativi di indice 0, 1) e `src` rappresenta la sorgente all'interno della CPU da cui viene estratto il dato da scrivere in memoria.

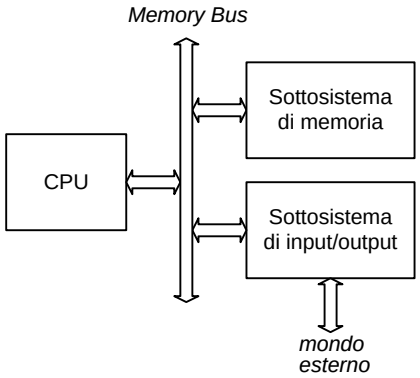


Fig. 32 Esistono delle CPU, come ad esempio quelle di tipo RISC (*Reduced Instruction Set Computer*), che, a differenza del PD-32, non dispongono di un bus di Input/Output e di conseguenza neanche delle relative istruzioni. In casi di questo genere, la CPU comunica col sottosistema di I/O con la tecnica detta *Memory Mapped I/O*, in base alla quale le "porte" di Input/Output residenti sulle interfacce vengono trattate come se fossero delle normali locazioni di memoria; naturalmente, l'assegnazione degli indirizzi deve avvenire in maniera tale da evitare conflitti di accesso tra il sottosistema di memoria e il sottosistema di I/O: in altri termini, il sottosistema di memoria *non* contiene tutta la memoria fisica indirizzabile dalla CPU.

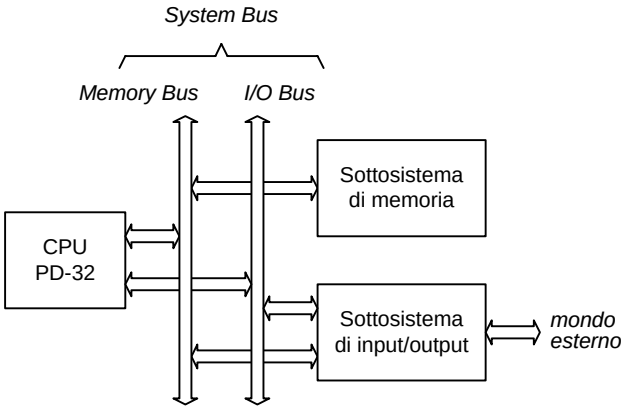


Fig. 33 La tecnica del Memory Mapped I/O può essere applicata anche al PD-32 e alle altre CPU dotate di I/O Bus; può addirittura essere utilizzata anche una tecnica mista, in cui il sottosistema di I/O è costituito da interfacce che operano soltanto in I/O standard, oppure soltanto in Memory Mapped I/O, oppure in entrambe le modalità. Naturalmente, nel secondo caso l'interfaccia sarà attestata solo sul Memory Bus, mentre nel terzo caso l'interfaccia sarà attestata sia sull'I/O Bus che sul Memory Bus. Poiché la CPU non ha alcun modo di sapere se la locazione di memoria a cui stia facendo riferimento risiede nel sottosistema di memoria oppure nel sottosistema di I/O, sarà cura delle interfacce memory mapped adattarsi ai protocolli previsti nei cicli di memoria generati dalla CPU e precedentemente descritti.

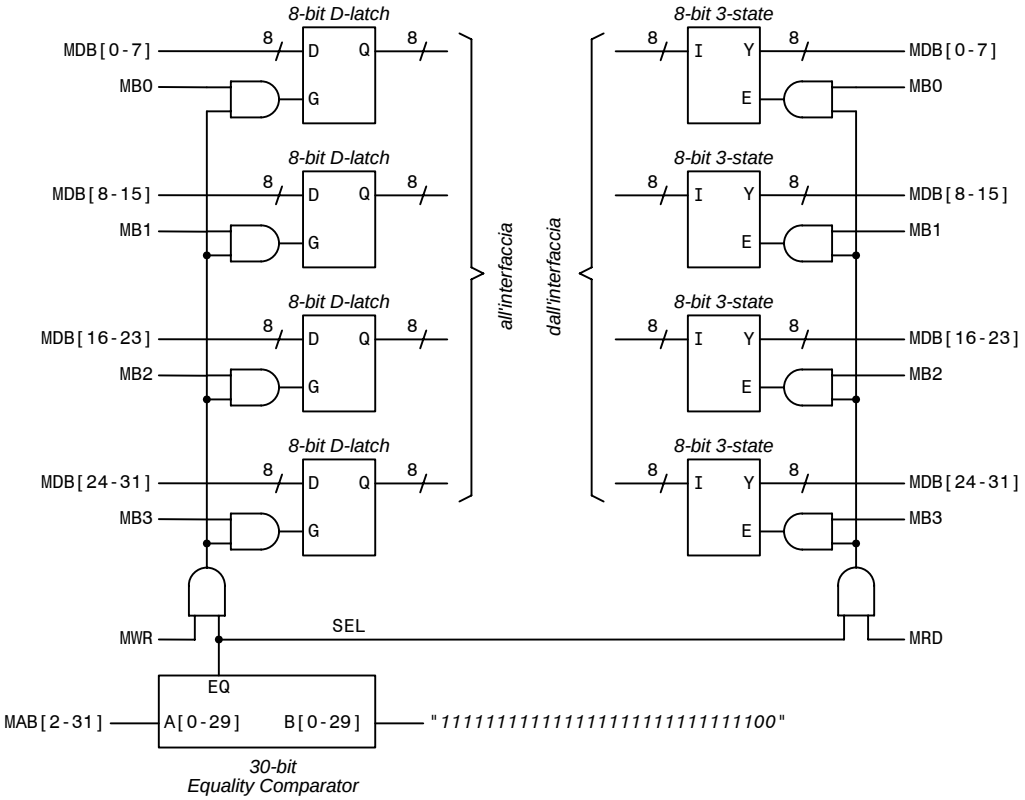


Fig. 34 Esempio di memory mapped I/O. Quattro “porte” di output e altrettante di input, da un byte ciascuna, sono allocate all’indirizzo `FFFFFFF0h`; la CPU può accedere ad esse come se fossero locazioni di memoria, e trattarle come “porte” da 8, da 16 o da 32 bit. Si noti come il segnale `SEL` per l’abilitazione delle “porte” venga in questo caso derivato dal Memory Address Bus anziché dall’I/O Address Bus, attraverso un Equality Comparator a 30 bit; si noti anche come ciascun byte delle “porte” sia abilitato dal corrispondente segnale di Memory Bank; si noti infine che, a differenza di una normale locazione di memoria e a somiglianza di quanto accade con le normali porte di I/O, il dato scritto su una “porta” con un dato indirizzo può essere differente da quello che verrà poi riletto dalla “porta” con lo stesso indirizzo.

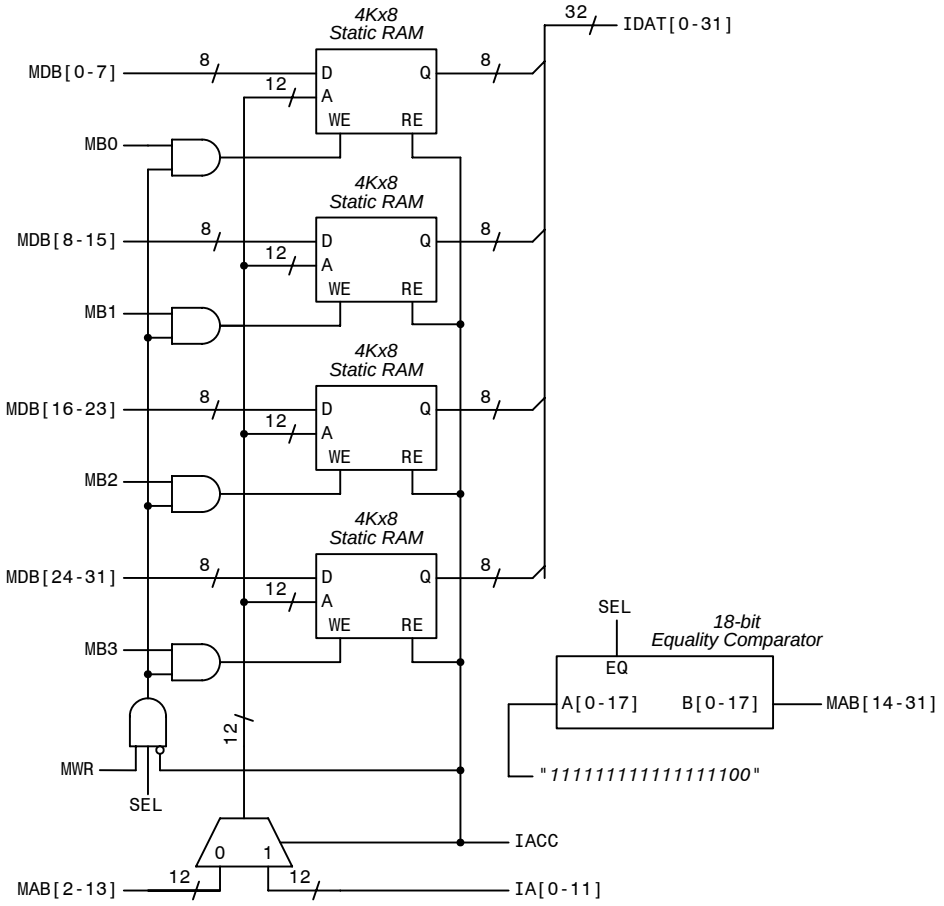


Fig. 35 Il Memory Mapped I/O non ha uno spazio di indirizzi così limitato come quello dell'I/O standard; tuttavia, quando il numero di "porte" supera un dato limite, può essere conveniente utilizzare delle RAM statiche, anziché array di registri o di buffer tri-state, per l'implementazione delle "porte". In tal caso l'insieme delle "porte" può essere trattato come un blocco di *memoria condivisa* (*shared memory*) tra CPU e interfaccia. A differenza delle normali memorie condivise, tuttavia, quasi mai nel memory mapped I/O l'accesso alle "porte" da parte della CPU e dell'interfaccia deve essere simultaneo: solitamente, l'interfaccia accede durante la normale operazione e, quando essa è *idle*, ossia in attesa di dati e/o comandi, la CPU accede alle "porte" per leggervi o scrivervi dati. Questa doppia modalità operativa delle "porte", indicata nello schema in figura dallo stato di un segnale `IACC` (*Internal Access*) generato dall'interfaccia (`IACC` = 1: accesso consentito all'interfaccia; `IACC` = 0: accesso consentito alla CPU), permette di semplificare notevolmente i circuiti delle "porte". In figura è illustrato un blocco di 16384 "porte" di output da 8 bit ciascuna, allocato all'indirizzo di memoria `FFFF0000h`, dove le porte sono realizzate mediante 4 moduli RAM da 4K×8, ciascuno attestato su una porzione del Memory Data Bus. Quando `IACC` = 0, la CPU può accedere in scrittura alle RAM utilizzando normali cicli di Memory Write; quando `IACC` = 1, l'accesso in scrittura è inibito e l'interfaccia, mediante un indirizzo $IA_{0\text{-}11}$ (*Internal Address*) generato internamente e applicato alle RAM mediante un multiplexer, vede i moduli RAM come un array di 4096 registri da 32 bit ciascuno.

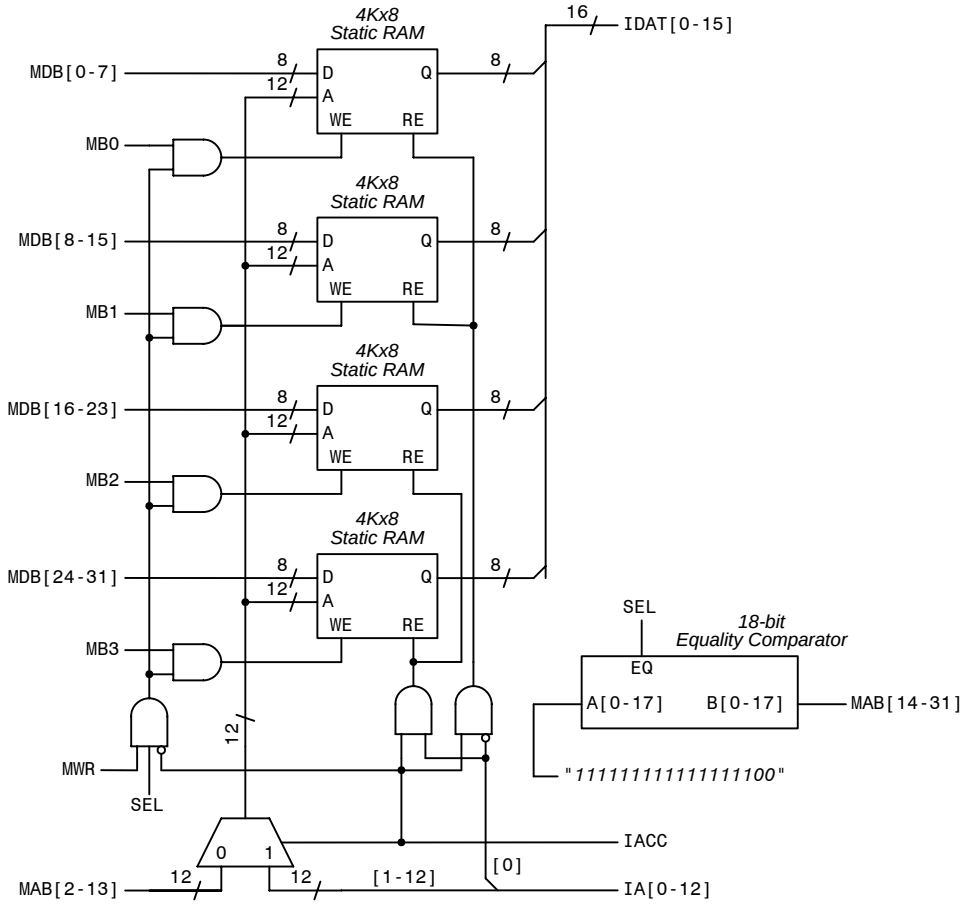


Fig. 36 Lo stesso blocco di 16384 “porte” di output da 8 bit già illustrato in Fig. 35; in questo caso, tuttavia, l’interfaccia vede il blocco come un array di 8192 registri da 16 bit, attraverso un indirizzo interno $\texttt{IA}_{0\text{-}12}$ a 13 bit: quando `IACC` = 1, i 12 bit più significativi di `IA` vengono utilizzati come indirizzi per i moduli RAM, e il bit meno significativo ($\texttt{IA}_0$) è utilizzato per attivare in lettura i due moduli RAM superiori oppure i due moduli RAM inferiori.

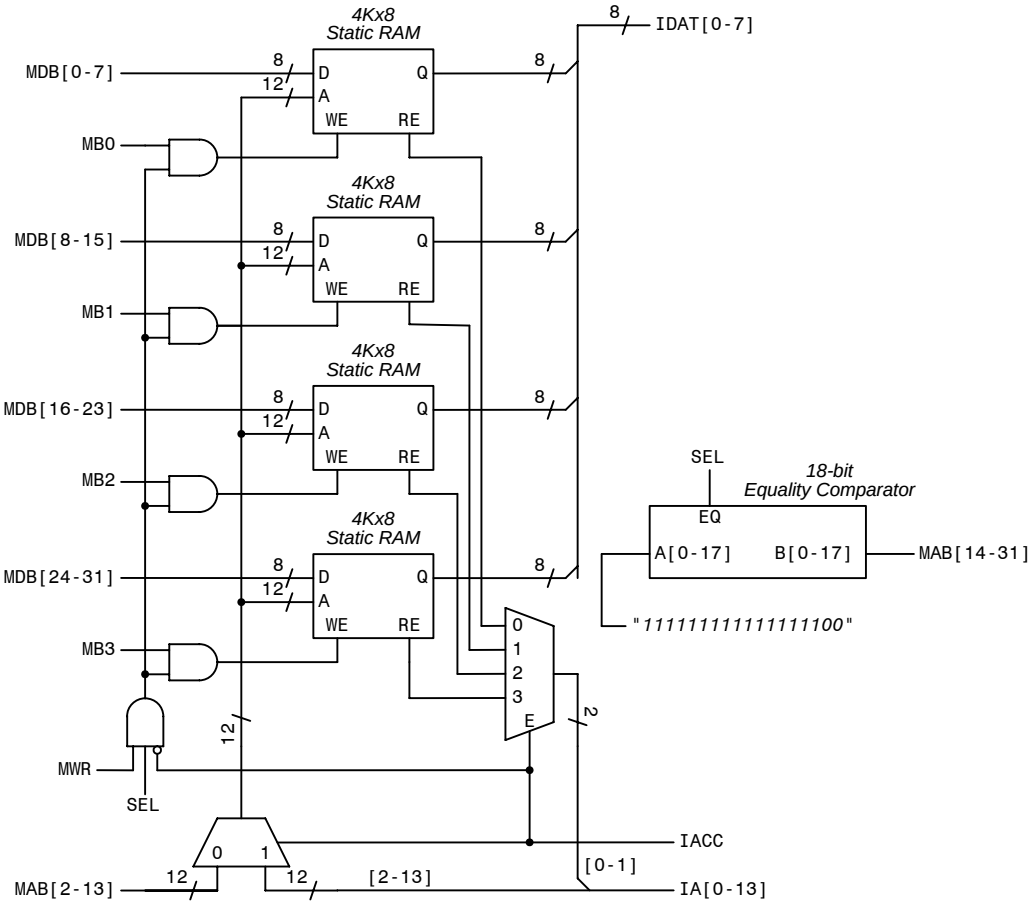


Fig. 37 Lo stesso blocco di 16384 “porte” di output da 8 bit già illustrato in Fig. 35; in questo caso, tuttavia, l’interfaccia vede il blocco come un array di 16384 registri da 8 bit, attraverso un indirizzo interno $IA_{0\text{-}13}$ a 14 bit: quando IACC = 1, i 12 bit più significativi di IA vengono utilizzati come indirizzi per i moduli RAM, e i due bit meno significativi ($IA_{0\text{-}1}$) sono utilizzati per attivare in lettura, attraverso un decoder a 4 uscite, uno solo dei moduli RAM per volta.

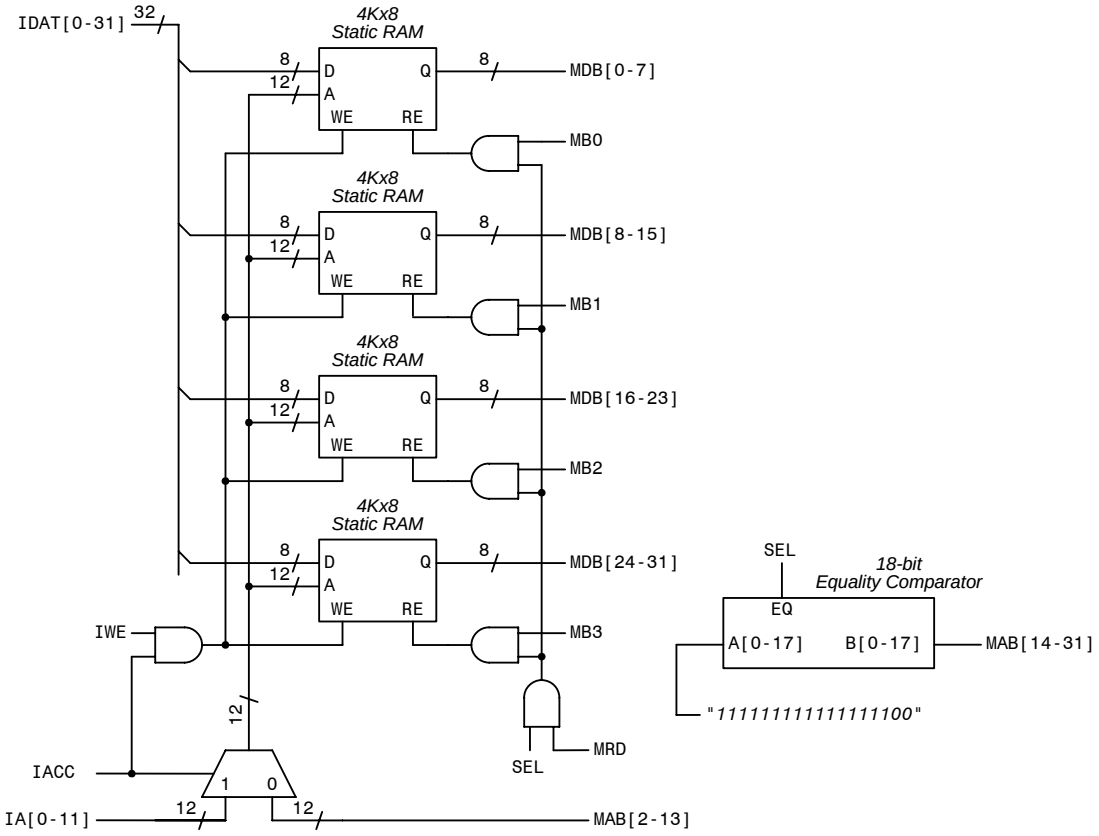


Fig. 38 Un blocco di 16384 “porte” di input a 8 bit ciascuna, mappate in memoria all’indirizzo `FFFF0000h`, realizzate con 4 moduli di RAM statica 4K×8. Quando `IACC` = 0, l’accesso al blocco è permesso solo alla CPU per la lettura mediante normali cicli di Memory Read; quando `IACC` = 1, la lettura da CPU è inibita e l’interfaccia può accedere al blocco RAM, vedendolo come un array di 4096 registri da 32 bit ciascuno indirizzato mediante $IA_{0\text{-}11}$ generato dall’interfaccia. La scrittura avviene in presenza di un apposito segnale di abilitazione (*Internal Write Enable*, `IWE`), anch’esso generato dall’interfaccia.

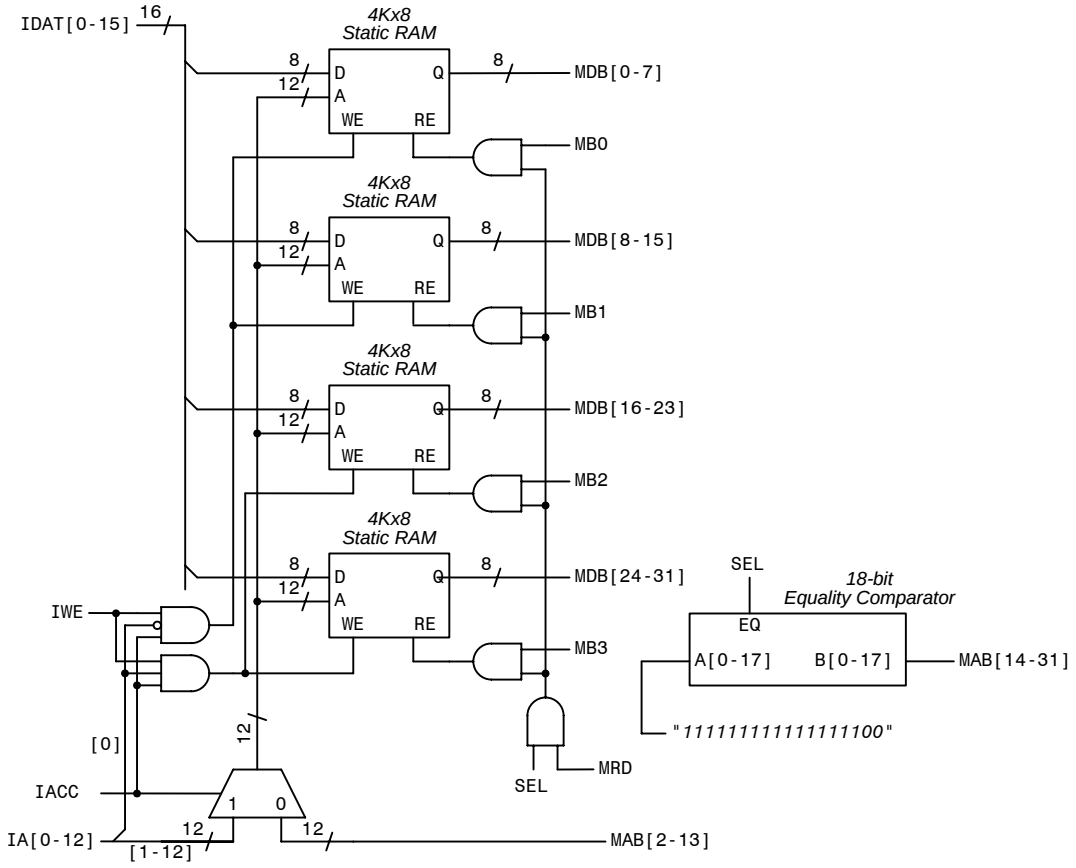


Fig. 39 Lo stesso blocco di 16384 “porte” di input mappate in memoria, già illustrato in Fig. 38; in tal caso, però, l’interfaccia vede il blocco come un array di 8192 registri da 16 bit ciascuno.

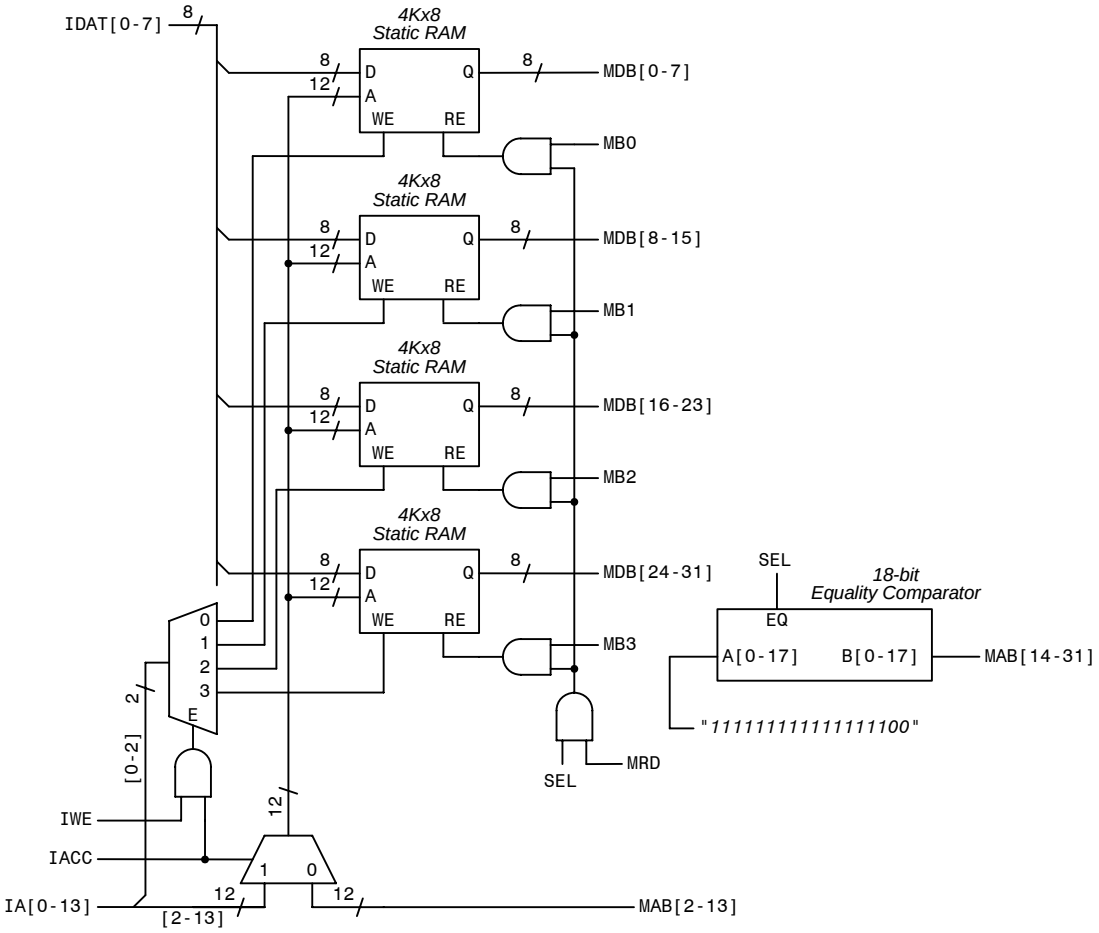


Fig. 40 Lo stesso blocco di 16384 “porte” di input mappate in memoria, già illustrato in Fig. 38; in tal caso, però, l’interfaccia vede il blocco come un array di 16384 registri da 8 bit ciascuno.