Il bus VME (IEEE standard 1014-1987/IEC 821 bus standard)

Il VME (Versa Module Europe) è l'erede del lavoro fatto dai progettisti della Motorola, a partire dal 1978, per dotare le CPU della serie 68k di un bus per il collegamento dei periferici all'altezza delle prestazioni che queste potevano esprimere. Questo lavoro produsse nel 1979 le prime specifiche di un "***crate bus***" "***proprietario***" denominato **Versabus**, successivamente accettato come standard (IEEE 970). La Motorola aveva bisogno di promuovere la produzione del suo microprocessore single chip sul quale aveva fatto enormi investimenti. Sperava con esso di entrare nel ricco mercato dei "mini-computer" che era in quel momento dominato dai PDP11 della Digital Equipment Corporazion (DEC) e dai 2100 della Hewlett & Packard e cominciò subito a produrre sia moduli CPU che moduli slave per il Versabus. Dopo un po' anche alcune altre ditte, collegate alla Motorola, come la Plessey Microsystems, inglese, cominciarono a produrre moduli compatibili col Versabus ma il successo dei sistemi basati sul Versabus fu molto condizionato, specialmente in Europa, dall'ineguatezza degli strumenti software.

In quegli stessi anni un organismo sovranazionale, l'International Electro-technical Commission (IEC) aveva proposto uno standard meccanico modulare per il formato delle schede elettroniche detto comunemente "***Eurocard***". Questo proposta era nata mettendo insieme lo standard di connettori multipolari della famiglia DIN 41612/IEC 603-2 ed i cassetti modulari definiti dallo standard IEEE 1101 PC/DIN 41494.

La Motorola decise di approfittare dell'incoraggiamento che la Comunità Europea dava in quel periodo a tutte le iniziative di standardizzazione in campo elettronico, aderendo con alcuni progettisti della sede di Monaco di Baviera ad un gruppo di lavoro per la definizione di un sistema di elaborazione distribuita, al quale diedero supporto anche la Mostek, la Signetics del gruppo Philips e la Thomson CSF.

Il gruppo di lavoro propose un sistema, in parte logicamente ispirato al Versabus, ma alloggiato in una meccanica Eurocard. Della parentela col Versabus rimase traccia nella sigla **VME**. La presenza nel gruppo di lavoro di progettisti non direttamente legati alla Motorola produsse, però, delle specifiche sufficientemente generali da poter essere interpretate efficacemente anche con CPU di altre marche. La veridicità di questa affermazione è oggi avvalorata dalla presenza sul mercato di molte schede processore in standard VME non basate su CPU Motorola. Sono adoperate CPU della Intel, della AMD, della HP, della Digital, della Sun e di altre marche ancora.

Quasi subito, sull'onda del successo dello standard, nacque, sotto gli auspici della IEEE, una associazione dei costruttori di moduli VME che si chiamò VMEbus International Trade Association (**VITA**). Questa organizzazione stampa due volte all'anno un'edizione aggiornata del catalogo di tutte le schede in standard VME e negli altri standard collegati (VXI e Futurebus+) prodotte nel mondo.

Il VME è un crate-bus, come il CAMAC ed il Fastbus. Quest'ultimo, però, ammette anche l'implementazione di segmenti su cavo (cable-segment).

La prima versione del VME (D16/A24) prevedeva schede Eurocard di altezza 3U (100 mm) o 6U (233 mm) con un connettore DIN 41612 a 96 pin su tre colonne: a, b e c. Il connettore, che nel formato 6U era montato in alto nella scheda (P1), prevedeva 16 bit per i dati e 24 bit per gli indirizzi. Non è casuale la perfetta corrispondenza con le caratteristiche dell'MC 68.000.
La versione successiva (A32/D32) ha imposto il formato 6U utilizzando 32 dei pin del connettore DIN 41612 a 96 pin in posizione P2, e lasciando a disposizione dell'utente i rimanenti 64 pin. La revisione "C.3" del 1987 è stata in uso per molti anni prima di essere sostituita dalla revisione D che è stata in vigore per poco tempo perché è stata subito introdotta una estensione a 64 bit del bus, detta VME64.

Le caratteristiche sintetiche del bus VME sono graficamente riassunte nel diagramma a blocchi funzionali di figura 1 (pag.2).

Nelle tabelle 1 e 2 sono date le assegnazioni dei pin sulle 3 file di contatti dei due connettori, il P1/J1 ed il P2/J2 previsti sulla scheda 6U (A32/D32).

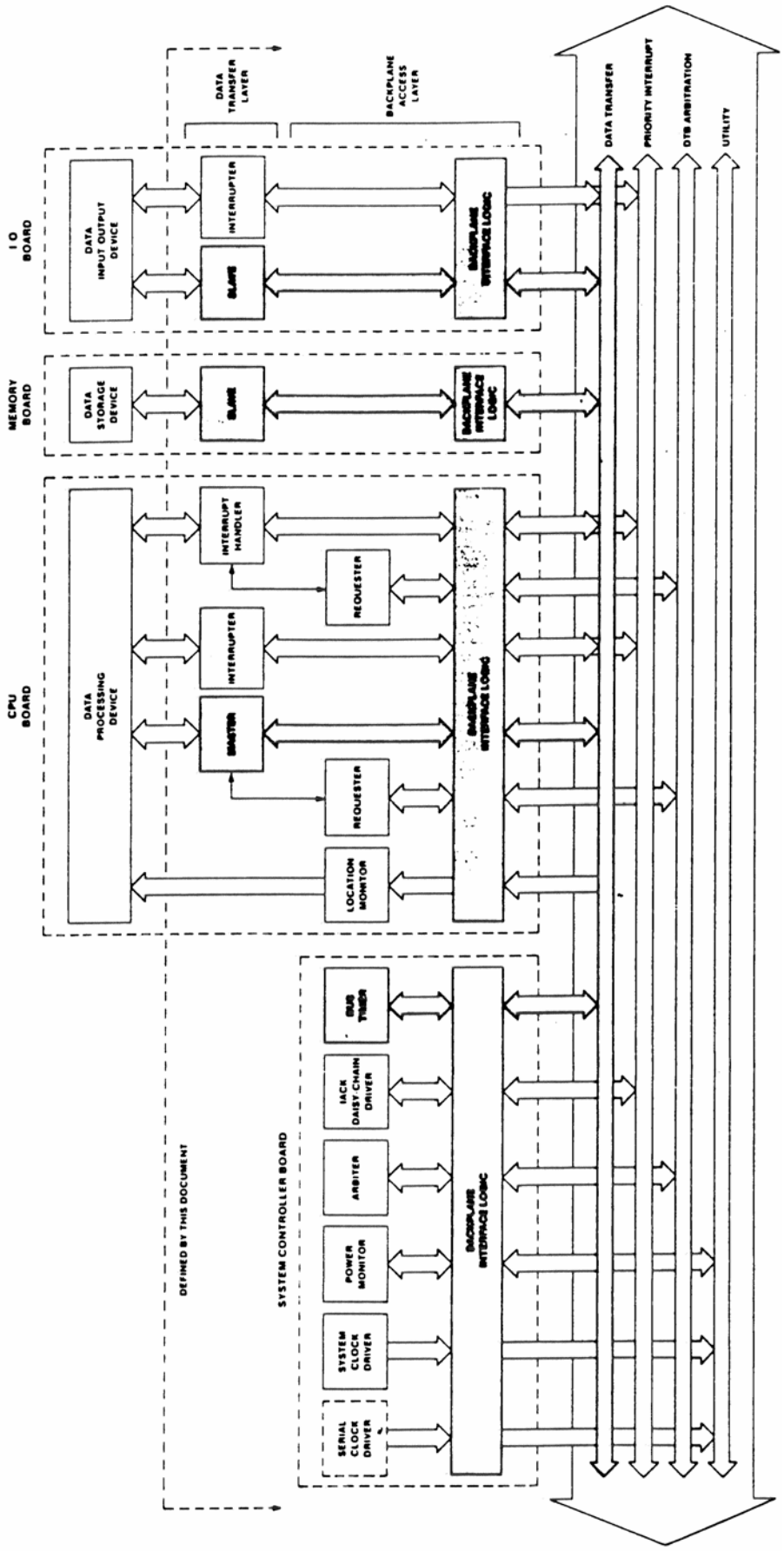

DATA TRANSFER LAYER
BACKPLANE ACCESS LAYER
I O BOARD
DATA INPUT OUTPUT DEVICE
INTERRUPTER
SLAVE
BACKPLANE INTERFACE LOGIC
MEMORY BOARD
DATA STORAGE DEVICE
SLAVE
BACKPLANE INTERFACE LOGIC
CPU BOARD
DATA PROCESSING DEVICE
INTERRUPT HANDLER
REQUESTER
INTERRUPTER
MASTER
BACKPLANE INTERFACE LOGIC
REQUESTER
LOCATION MONITOR
DEFINED BY THIS DOCUMENT
SYSTEM CONTROLLER BOARD
BUS TIMER
IACK DAISY CHAIN DRIVER
ARBITER
POWER MONITOR
SYSTEM CLOCK DRIVER
SERIAL CLOCK DRIVER
BACKPLANE INTERFACE LOGIC
DATA TRANSFER
PRIORITY INTERRUPT
DTB ARBITRATION
UTILITY

Fig. 1

Le specifiche VME per favorire una migliore comprensione della struttura del bus suddivide l'insieme delle linee, dal punto di vista funzionale in cinque "sub-busses" ciascuno in grado di eseguire autonomamente un certo tipo di operazione specifica. Questa suddivisione è evidenziata in figura 1 dove però sono indicati solo quattro dei gruppi di linee e cioé, il **bus trasferimento dati**, il **bus per l'arbitraggio**, il **bus per la gestione delle priorità di interruzione** ed il **bus dei servizi**.

Quello che manca, il quinto, è uno speciale bus seriale detto **VMSbus** (**VME Message Serial bus**) che può essere utilizzato per scambiare brevi messaggi urgenti tra moduli o sistemi VME. Esso utilizza due soli pin SERCLK e SERDAT* (Tabella 1).

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02	ACFAIL*	D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	SYSFAIL*
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	LWORD*
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*	SERCLK (1)	A17
22	IACKOUT*	SERDAT* (1)	A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31	-12V	+5V STDBY	+12V
32	+5V	+5V	+5V

(1) VMS BUS SIGNALS

Tab. 1

L'assegnazione delle linee agli altri quattro bus nel VME A32/D32 è la seguente:
"**Data Transfer bus**", utilizza tre diversi tipi di linee:

Indirizzamento: A01-A31, AM0-AM5, DS0*, DS1*, LWORD*.

Dati: D00-D31.

Controlli: AS*, DS0*, DS1*, BERR*, DTACK*, WRITE*.

Il "**Priority Interrupt bus**" utilizza le linee:
IRQ1*-IRQ7*, IACK*, IACKIN*, IACKOUT* , nonché quelle A01-A03, AS*, DS0*, DS1*, LWORD* e D00-D31, DTACK*.
"**Data transfer arbitration bus**", utilizza le linee:
BR0*-BR3*, BG0IN*-BG3IN*, BG0OUT*- BG3OUT*, BBSY*, BCLR*.
Lo "**Utility bus**" raggruppa quattro linee:
SYSCLK*, SYSRESET*, SYSFAIL ed ACFAIL*.

Le funzioni di ciascuno dei bus sono chiaramente indicate dal rispettivo nome.

Il modello di architettura logica utilizzato, a tutti i livelli, è quello master/slave, i trasferimenti sul bus sono asincroni e le linee dati e quelle indirizzi sono separate.

I campi d'indirizzamento possono utilizzare 16-bit (short), 24-bit (standard) e 32-bit (extended), con possibilità di passare dall'uno all'altro dinamicanente, così come si possono gestire dinamicamente, byte (8-bit), word (16-bit) e long word (32-bit).

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	USER I/O	+5 Volts	USER I/O
2	USER I/O	GND	USER I/O
3	USER I/O	RESERVED	USER I/O
4	USER I/O	A24	USER I/O
5	USER I/O	A25	USER I/O
6	USER I/O	A26	USER I/O
7	USER I/O	A27	USER I/O
8	USER I/O	A28	USER I/O
9	USER I/O	A29	USER I/O
10	USER I/O	A30	USER I/O
11	USER I/O	A31	USER I/O
12	USER I/O	GND	USER I/O
13	USER I/O	+5 Volts	USER I/O
14	USER I/O	D16	USER I/O
15	USER I/O	D17	USER I/O
16	USER I/O	D18	USER I/O
17	USER I/O	D19	USER I/O
18	USER I/O	D20	USER I/O
19	USER I/O	D21	USER I/O
20	USER I/O	D22	USER I/O
21	USER I/O	D23	USER I/O
22	USER I/O	GND	USER I/O
23	USER I/O	D24	USER I/O
24	USER I/O	D25	USER I/O
25	USER I/O	D26	USER I/O
26	USER I/O	D27	USER I/O
27	USER I/O	D28	USER I/O
28	USER I/O	D29	USER I/O
29	USER I/O	D30	USER I/O
30	USER I/O	D31	USER I/O
31	USER I/O	GND	USER I/O
32	USER I/O	+5 Volts	USER I/O

Tab. 2

Il bus prevede una linea di errore (BERR*) per il riconoscimento di situazioni impreviste ed ha un limite superiore teorico di trasferimento dati di 40 Mbyte/sec che deriva dall'ipotesi che si possano effettuare attraverso il backplane trasferimenti di long word (4 bytes) a 10 MHz.

Il bus ha 7 livelli di priorità per le interruzioni. Le potenzialità del sistema di arbitraggio gli consentono di gestire anche un crate di tutte schede processori (21) ed ha una linea che consente la diagnostica dei guasti, SYSFAIL*.

Una caratteristica particolare del bus VME sono le 6 linee di "**address modifier code**", AM0-AM5 che sono alla base della duttilità di indirizzamento e di gestione dati nel sistema. Ad essi è anche affidata l'informazione sulla modalità del trasferimento che, come nelle CPU 68k, può essere "***supervisory***" e "***non-privileged***". In tabella 3 è data l'assegnazione dei codici degli address modifier, sottolineando che il loro uso è in alternativa alla linea IACK* perchè durante un ciclo di interrupt acknowledge 3 delle 32 linee indirizzo, A01, A02 ed A03 sono impegnate e non vi può essere un ciclo di indirizzamento di alcun tipo.

A.M. (hex)	**IACK***	**Address bits**	**Tipo di trasferimento**
3F	1	24	Standard supervisory block transfer
3E	1	24	Standard supervisory program access
3D	1	24	Standard supervisory data access
3B	1	24	Standard non-privileged block transfer
3A	1	24	Standard non-privileged program access
39	1	24	Standard non-privileged data access
2D	1	16	Short supervisory access
29	1	16	Short non-privileged access
10-1F	1	—	Definibili dall'utente
0F	1	32	Extendend supervisory block transfer
0E	1	32	Extendend supervisory program access
0D	1	32	Extendend supervisory data access
0B	1	32	Extendend non-privileged block transfer
0A	1	32	Extendend non-privileged program access
09	1	32	Extendend non-privileged data access
XX	0	3	Interrupt acknowledge cycle

Tab. 3

In figura 2 è mostrato il diagramma delle temporizzazioni delle singole operazioni sulle varie linee fatte da un master e da uno slave durante un tipico ciclo di lettura dati in uno slave (Read cycle). L'esempio si riferisce ad un caso in cui si usa la tecnica dello "address pipelining", perchè il nuovo indirizzo è già sulle apposite linee quando il primo ciclo di read ancora non è finito.
Gli intervalli di tempo che vengono recuperati, sottraendoli ai laschi previsti dal protocollo sono indicati in figura come Address rot e Data rot.

La successione delle operazioni è la seguente:

1) Il master indirizza il periferico attivando le linee A01-A32, AM0-AM5, IACK* ed LWORD* (si tratta di un trasferimento di 32 bit). La linea WRITE* deve essere negata trattandosi di un read.

2) Il master convalida le linee indirizzi con la transizione verso il livello basso della linea AS* e mette basse la linea DS0* e/o quella DS1*. Attenzione che queste linee nei diagrammi temporali sono indicate con DSA* e DSB* per indicare che tra di esse può esserci uno "skew" (differenza di tempi di propagazione) apprezzabile.
3) Lo "slave" decodifica l'indirizzo, abilita in uscita il suo registro dati (mette i dati sul bus) e ne da comunicazione al master "asserendo" la linea DTACK*.
4) Quando il master ha regitrato i dati nel suo latch lo comunica allo "slave" negando la linea o le linee di data strobe
5) Lo slave risponde togliendo il dato dal bus e negando la linea DTACK*. Questa operazione chiude il ciclo, ma, come si vede dal diagramma, se questo non era l'ultimo della serie, sulle linee indirizzo c'è già un nuovo indirizzo.

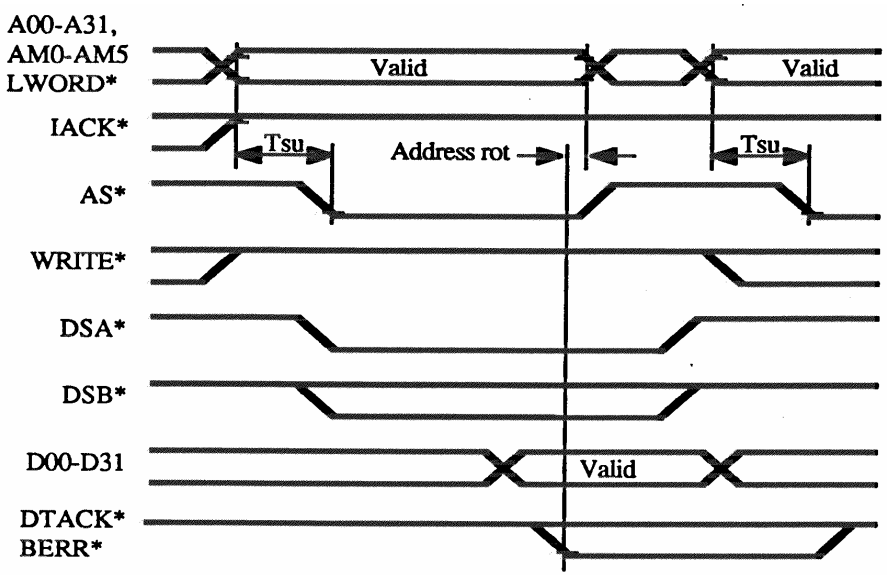


Fig. 2

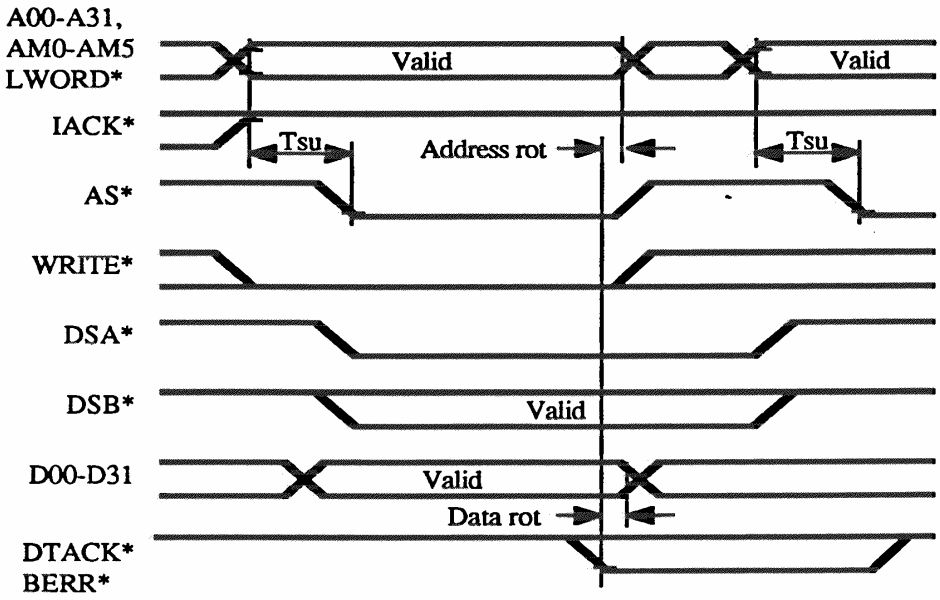


Fig. 3

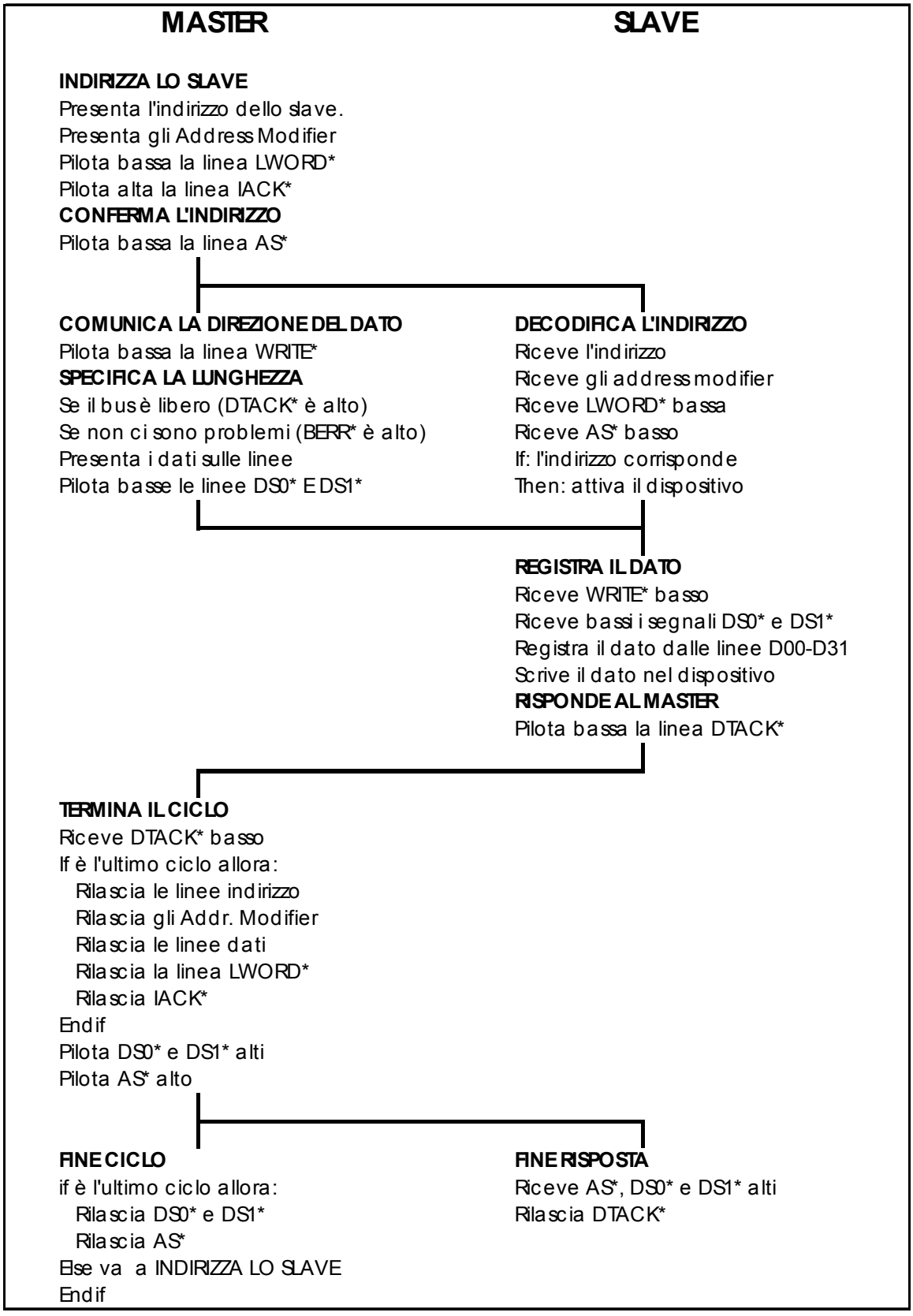


Fig. 4

Da quanto abbiamo visto si può chiaramente dedurre che le specifiche del VME hanno dovuto proporre tecniche completamente inedite per definire lo standard. Per dare, infatti, un'idea, anche solo indiretta della complessità di un bus multiprocessore come il VME, è sufficiente dire che è stato necessario proporre perfino una specifica terminologia basata su 5 parole chiave: REGOLE,

RACCOMANDAZIONI, SUGGERIMENTI, PERMESSI ed OSSERVAZIONI. Il significato è abbastanza trasparente ma la cosa importante è che nelle specifiche ogni affermazione deve ricadere in uno di questi 5 casi.

L'arbitraggio del bus è un problema cruciale in un bus multi-master e lo è ancora di più in un sistema di elaborazione distribuita, in cui le esigenze dei vari master, che aspirano a diventare commander del bus per scambiare dati con gli slave, si intrecciano con i processi software che sono in esecuzione nei master intelligenti.

Le specifiche VME hanno affrontato il problema con estrema attenzione ed hanno operato in due direzioni diverse. Da una parte prevedono l'uso di collegamenti tra schede processori e periferici (in particolare memorie) su bus ausiliari, paralleli, come ad esempio, il VSB o, seriali, come il VMS, per rendere la gestione di ogni sottosistema il meno critica possibile, dall'altra prevedono una struttura hardware per l'arbitrazione del bus molto duttile e potente ma che possa, all'occorrenza, essere facilmente minimizzata per non appesantire i piccoli sistemi che non hanno bisogno di una gestione sofisticata delle priorità per assumere il possesso del bus.

Il bus arbitrazione prevede quattro distinti livelli di richiesta del bus. Ciascun livello ha una propria linea di richiesta (BRX*) alla quale è associata una coppia di linee per la gestione del "bus grant" BGXIN*/BGXOUT* secondo la tecnica "daisy-chain". Che, come è noto, in ciascun livello assegna automaticamente la priorità al dispositivo più vicino all'arbitro che è supposto essere nella postazione 1 del crate. (Fig. 5)

Ciascuna linea di richiesta è associata alla coppia di linee di "bus grant" con lo stesso numero ed il livello 3 è, per convenzione, quello a più alta priorità mentre la priorità più bassa tocca al livello 0.

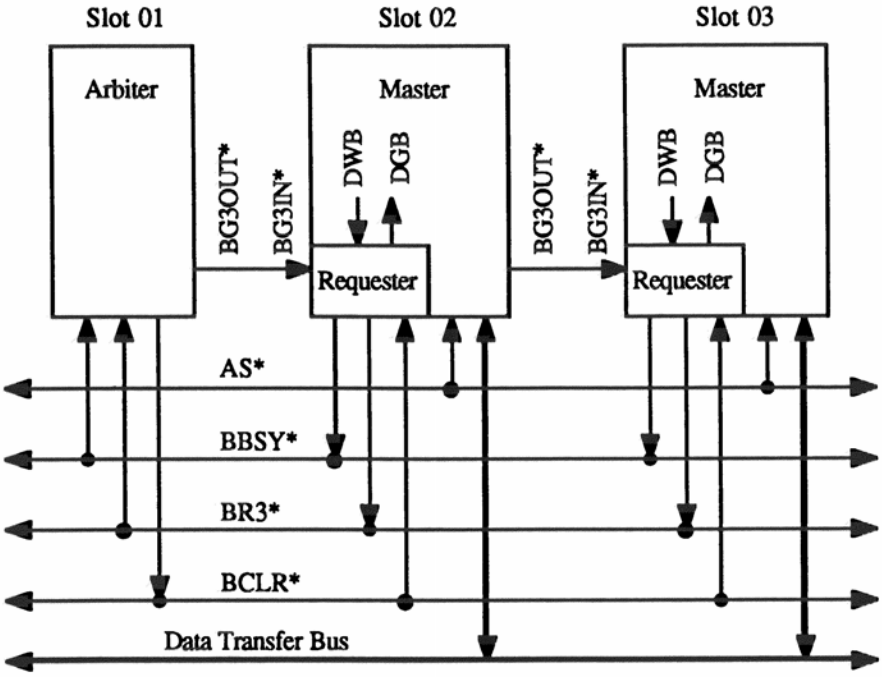


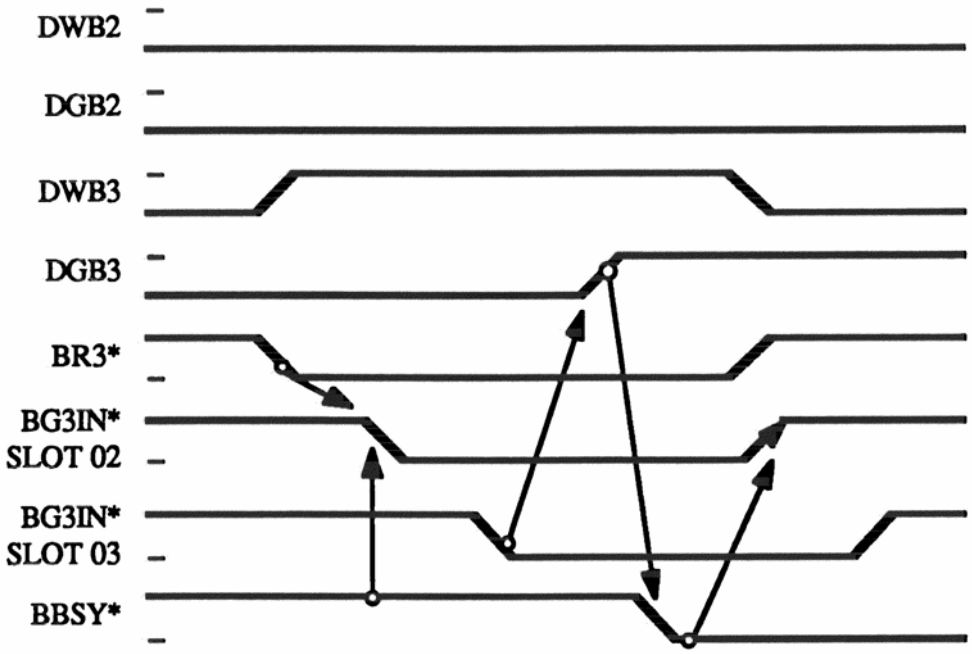
Fig. 5

Il funzionamento di ciascun livello è quello classico. La linea di richiesta è un wired-OR in logica negata con dispositivi open-collector. Quando la linea va bassa, l'arbitro, se il bus non è occupato (BBSY* attivo) genera un segnale di bus grant sulla linea corrispondente a quel livello di richiesta. La logica di richiesta del primo dispositivo lungo la linea di grant, riceve il segnale e, se non aveva attivata bassa la linea di richiesta, lo propaga al successivo, e così di seguito, fin che il segnale non arriva al controllore dell'interrupt del dispositivo richiedente, il quale, non propaga oltre il grant ed attiva, invece, la linea di busy, dopo aver comunicato al proprio master di avere assunto il controllo del bus attraverso il segnale DGB. In sostanza questo segnale chiude il ciclo di "hand-

shaking" tra Requester e Master che era cominciato col segnale di DWB, il cui nome, probabilmente, deriva da: "Device Want Bus".

Il dispositivo che in un certo momento controlla il bus può tenerlo indefinitamente, a meno che, l'arbitro non attivi la linea di BCLR*. La logica di attivazione della linea dipende dal metodo di arbitrazione, ma non è possibile attivarla se il bus non è occupato (BBSY* basso).
All'interno delle specifiche sono suggerite varie tecniche di priorità che rispondono a differenti requisiti.

Si può usare una tecnica gerarchica che rischia di escludere dalla possibilità di ottenere il controllo del bus i dispositivi che si trovino ad un livello di priorità inferiore, rispetto a dispositivi che impegnano frequentemente il bus.



(b) Timing Waveform.

Fig. 6

Si può usare una tecnica di rotazione (Round-Robin Arbiter) che garantisce a ciascun dispositivo, a rotazione, un tempo massimo, dopo il quale, se ci sono altre richieste, viene attivato il BCLR*.

Ogni sistema VME deve avere un arbitro del bus e questo deve risiedere nella postazione 01 del bus (estrema sinistra del crate).

Questo obbligo non costituisce un problema perchè ormai tutte le schede CPU hanno al loro interno almeno "un single level arbiter" cioè un arbitro minimale che gestisce con la tecnica "daisy-chain" solo il livello 3.

Un'altro punto fondamentale di un bus per applicazioni "real time" è il sistema di interrupt. Abbiamo già visto che il VME prevede ben 7 livelli di priorità per il program interrupt. Un ciclo di interruzione comincia con l'attivazione da parte di uno slave di una delle 7 linee di richiesta che, essendo strutturate come linee omnibus, possono essere utilizzate da un numero qualsiasi di dispositivi.

Questo ciclo ha due importanti funzioni, una è l'arbitraggio dell'interrupt e l'altra è il ciclo di lettura dello STATUS/ID (vettore di interrupt). Quest'ultima operazione implica che il modulo di gestione dell'interrupt sia in grado di richiedere ed ottenere il controllo del bus dati attraverso il quale potrà ricevere dallo slave il vettore di interruzione che servirà ad individuare in memoria la sua "routine di servizio". Questa procedura elimina la necessità del polling per trovare il dispositivo richiedente ed associarlo ad una routine di servizio.

Tutto questo si esprime sinteticamente attraverso l'affermazione di gergo specialistico che il VME prevede una procedura di **"Interrupt vettorizzato"**.

Quando l'interrupt handler VME avrà ottenuto il bus dati piloterà le linee A01-A03, attivando basse le linee IACK* e AS*. Sulle linee indirizzi viene messo il codice binario del livello di priorità corrispondente alla linea da cui è arrivata la richiesta e viene confermato dalla linea AS*. Gli A.M. non debbono essere usati.

Questa operazione serve ai dispositivi che operano con un diverso livello di priorità per non interferire nella fase successiva. Essi, riscontrando che il loro livello di priorità è diverso da quello che sta per essere servito, dovranno rendersi "trasparenti" al segnale che punta ad individuare il dispositivo richiedente, anche se nel frattempo avevano richiesto, a loro volta, l'Interrupt.
Il modulo di gestione (interrupt handler) tiene sotto osservazione le linee di richiesta e, quando rileva una linea attivata, chiede il controllo del bus. Dopo averlo ottenuto, in risposta al segnale, genera un "interrupt acknowledge cycle".

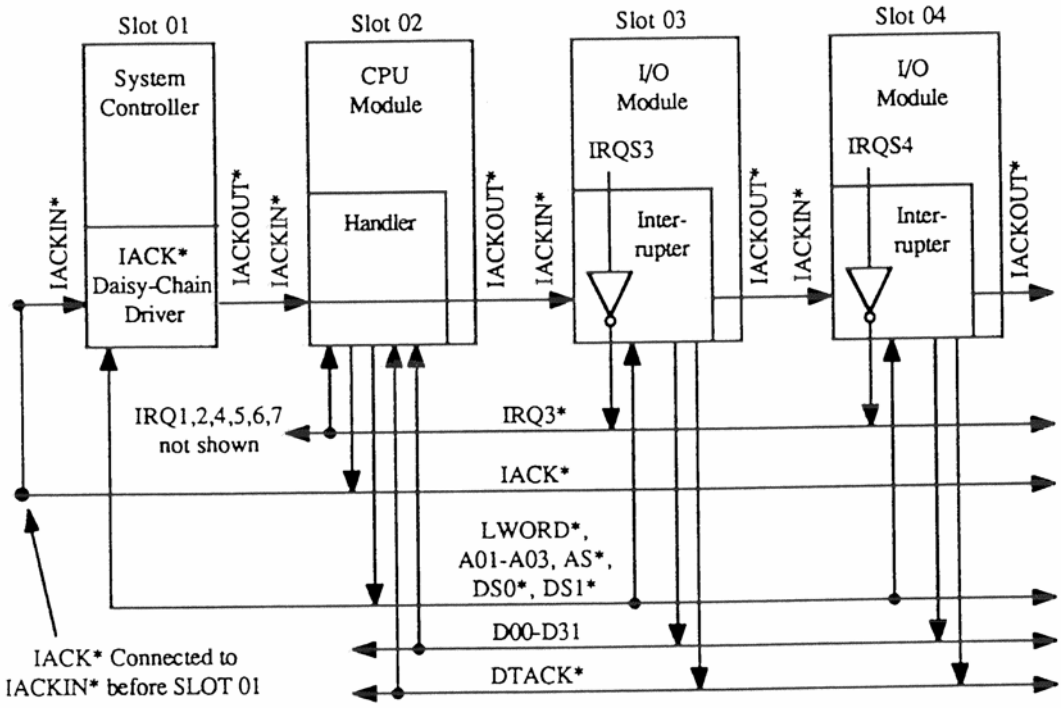


Fig. 7

In figura 7 è disegnato il diagramma a blocchi semplificato della struttura del bus delle interruzioni. Nello schema è stato ipotizzato che la richiesta sia venuta sulla linea IRQ3* e quindi durante il ciclo di "acknowledge" le linee A03-A01 saranno, nell'ordine a "0", "1", "1".

In figura 8 è dato il diagramma temporale del ciclo di interrupt acknowledge per un vettore d'interruzione (STATUS/ID) ad 8 bit.
Il segnale IACK* basso indica che è in atto un ciclo di interrupt.

L'arbitrazione dell'interrupt viene effettuata con la tecnica della daisy-chain che deve essere gestita da un driver che si trova nella slot 1 del bus. Esso, se non è anche interrupt handler, riceve da quest'ultimo il segnale IACK* sul pin IACKIN* e dopo aver ricevuto basso DS0* o DS1* od entrambi eventualmente accompagnati da LWORD* (a seconda che sia previsto un vettore di interrupt ad 8, 16 o 32 bit) lo propaga attraverso il pin IACKOUT* verso il pin IACKIN* della successiva slot occupata. Quando il modulo che ha richiesto l'interrupt (o il primo di quelli che hanno attivata la linea) riceve dal modulo che lo precede lungo il bus il segnale IACKIN*, presenta sulle linee dati il vettore STATUS/ID e termina il ciclo asserendo bassa la linea DTACK*.

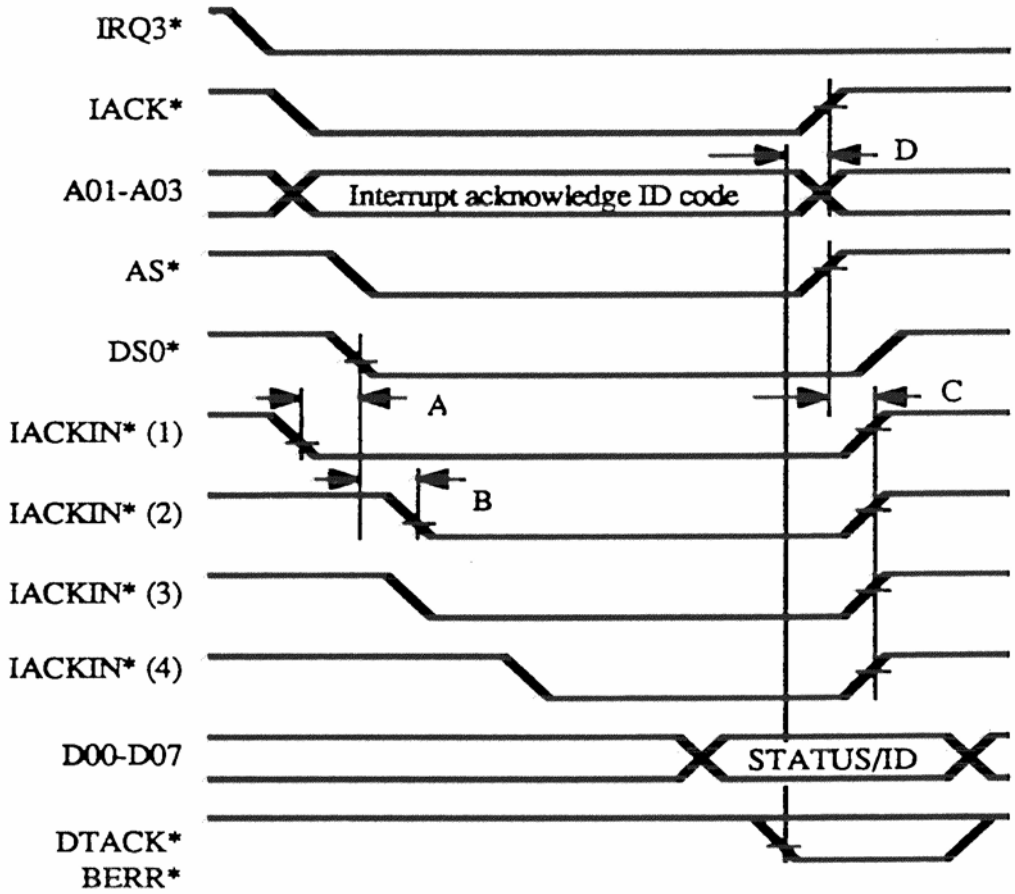


Fig. 8

Il segnale DTACK* avvia la procedura di rilascio da parte dell'handler che pilota alta le linee di IACK* e di AS* e rilascia le linee A01-A03. A questo punto i dispositivi sul bus attraversati dal segnale della daisy-chain, dopo che l'handler avrà negato la linea AS*, debbono, entro un tempo preciso (40 ns), pilotare alte le linee IACKIN*/ IACKOUT* per evitare che si intreccino due diversi cicli di interrupt.