di Andrea de Prisco

Cooperazione, Bus, Arbitraggio

Dopo i vari «interrupt» giocosi dei mesi scorsi, Appunti di Informatica riprende (finalmente) a parlare di calcolatori. L'argomento di questo mese riguarda la cooperazione tra le varie unità di cui un sistema di elaborazione è composto: parleremo di interfacciamento a basso livello, di collegamenti dedicati e ripartiti, di meccanismi di arbitraggio deterministici e non. Buona lettura.

Cooperazione

Ogni sistema di elaborazione è suddiviso, sia logicamente che fisicamente, in vari moduli che cooperano per portare a termine i vari «doveri» ai quali sono «chiamati». Mentre le silenziose CPU elaborano le istruzioni dei programmi in esecuzione, le stampanti martellano sul nastro inchiostrato per produrre i loro output così come il vecchio plotter, tra uno scricchiolio e l'altro, arranca un po' per portare a termine il disegno richiesto. Contemporaneamente il lampeggio delle lucette delle unità a dischi magnetici segnala l'avvenuto completamento dell'operazione di lettura o scrittura richiesta pochi centesimi di secondo prima.

Zoomando idealmente dentro una qualsiasi di queste unità «vedremmo» un interminabile brulichio di segnali elettrici tra i vari chip, che dialogano tra loro come il dispositivo che compongono dialoga con gli altri dispositivi della sala macchine...

Esistono essenzialmente due modi per far dialogare le varie componenti di un sistema di elaborazione: collegamenti dedicati o ripartiti. Col primo, per collegare due dispositivi si utilizza un collegamento fisico vero e proprio, nonché privato, tra i due, cosicché essi potranno scambiare dati in ogni momento, e senza dar conto ad altri. Lo schema di collegamento ripartito, più noto come collegamento a bus, a fronte di un risparmio di costo e a una espandibilità del sistema facilitata, contrappone la limitazione che un solo dispositivo alla volta può inviare messaggi ad altri. Ovvero, se due dispositivi nello stesso istante vogliono accedere al bus per dialogare con altri dispositivi, dovranno dapprima competere per l'accesso esclusivo alla struttura di interconnessione e poi, chi dei due sarà scelto (dall'arbitro), potrà iniziare il trasferimento dei dati. L'altro aspetta.

Per il momento lasciamo perdere bus e arbitri e torniamo ai collegamenti dedicati. In figura 1 abbiamo schematizzato l'interfaccia di un collegamento dedicato tra due unità. U1 e U2, dove la prima produce un dato e l'invia alla seconda: potrebbero essere ad esempio rispettivamente un proces-

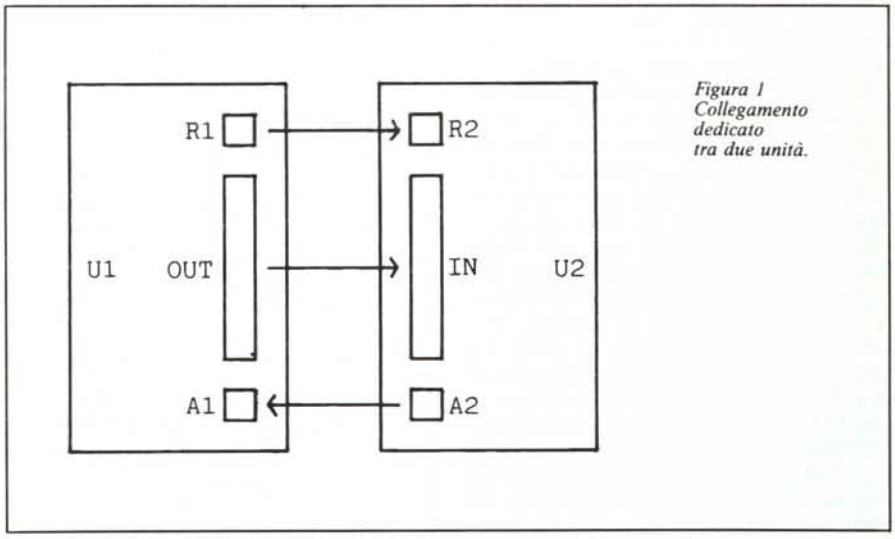

Figura 1 Collegamento dedicato tra due unità.

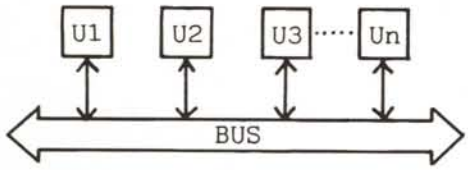

Figura 2 Schema di collegamento Bus.

sore di I/O e una stampante... ma potrebbero anche essere una CPU e una memoria o due computer completi. Ciò che dobbiamo realizzare è la sincronizzazione tra i due dispositivi: U2 deve prendersi il dato dall'interfaccia solo dopo che U1 glielo ha inviato, così come U1 non deve inviare altri dati se U2 non ha letto il dato precedente. L'interfaccia d'uscita di U1 è composta da un registro detto OUT atto a contenere il dato da inviare (formato quindi da un numero di bit pari alla lunghezza, sempre in bit, dei dati) e da due flip-flop (registri di un bit) che abbiamo chiamato R1 e A1. Dall'altro lato, U2 disporrà di un registro di ingresso IN, della stessa grandezza di OUT, più due flip-flop R2 e A2. Per la cronaca R sta per Ready, pronto, A per Acknowledgement, conoscenza. Tra queste tre coppie di registri, sempre come visibile in figura 1, i collegamenti elettrici non fanno altro che trasferire, nel verso della freccia, quanto presente in ognuno dei tre registri di uscita, nel corrispondente registro di ingresso del partner.

Commentiamo brevemente il funzionamento. Ricordiamo che U1 produce dati da inviare a U2. Appena un dato è pronto per essere inviato, U1 lo inserisce nel suo registro di uscita OUT e pone ad 1 lo stato di R1 (anch'esso registro di uscita), inizialmente a 0. In questo modo avverte U2 che il dato è pronto (Ready). Subito dopo, U1, resta in attesa che U2 dia «segni di vita». Dall'altro capo del collegamento, appena U2 è pronto a ricevere un nuovo valore testa innanzitutto il valore del suo registro di ingresso R2, se questo è ad 1 vuol dire che è presente un dato significativo nel suo registro IN, lo prende, e settando il valore di A2 (tenete sempre sott'occhio la figura 1) comunica a U1 di averlo ricevuto. Torniamo ad U1, il quale, come detto prima stava aspettando che U2 rispondesse (A2 e, conseguentemente, A1 pari a 1): ricevuta la risposta riazzera R1 per poi risettarlo al prossimo invio di un dato. U2 farà lo stesso col suo A2. Si noti che, come promesso, è realizzata la sincronizzazione tra i due dispositivi: non è possibile (per doveri... d'algoritmo) che né U1, né U2 avanzi senza esplicito consenso da parte del corrispondente partner.

Collegamenti ripartiti

In figura 2 è mostrato, molto a grandi linee, lo schema di collegamento ripartito, o a bus che dir si voglia. Come visibile, le varie unità sono tutte collegate alla medesima struttura di interconnessione detta appunto bus. Sul bus corrono i messaggi che le varie unità si scambiano durante il funzionamento di tutto il sistema. Si badi, come già detto prima, che per unità non si intende solo stampanti e drive per dischi magnetici, ma anche memoria, CPU, coprocessori ed altro. Per motivi di ottimizzazione, comunque, è assai difficile che le architetture reali si rifacciano completamente ad uno solo dei due schemi visti (interamente dedicato o interamente ripartito), ma si preferiscono normalmente i cosiddetti schemi misti, in cui per alcune unità si usano collegamenti dedicati, per altre uno o più bus di collegamento. In ogni caso, quando in un sistema esiste un collegamento ripartito, le unità che lo utilizzano possono accedere in scrittura solo una per volta. Tornando alla figura 2, immaginiamo che in un dato istante il bus sia libero: in altre parole nessuna unità sta dialogando con altre. Se in quel momento U1 decide di dialogare con U3 può farlo e lo schema di sincronizzazione non è molto dissimile da quello dedicato. L'unica differenza riguarda il fatto che nel messaggio da inviare oltre al «testo» vero e proprio, il mittente deve inserire anche il nome del destinatario. Ovvero alcuni bit del registro OUT dell'interfaccia di uscita di ogni unità contengono il codice del destinatario, i rimanenti il valore, come visto prima, da inviare. Il bus, ancora una volta, non è che un insieme di fili elettrici (uno per bit, compresi i sincronizzatori A e R, come prima) sui quali è presente l'informazione che chi scrive ha immesso nella sua interfaccia di uscita. Chi sta aspettando un messaggio dal bus, non dovrà fare altro che controllare se i primi bit corrispondono al suo «nome», nel qual caso può prelevare il dato e settare la linea A ad 1 per comunicare al mittente di aver ricevuto il dato.

A questo punto dovrebbe essere evidente che se due unità scrivono nello stesso istante qualcosa sul bus (che, lo ripetiamo, non è altro che un fascio di linee elettriche) i due messaggi si mischierebbero (destinatari, primi bit, compresi) rendendo impossibile il completamento delle due trasmissioni. Ovvero i due relativi destinatari dei due (distinti) messaggi sarebbero in grado di riconoscersi come tali essendo ormai il contenuto del bus non più significativo.

Meccanismi di arbitraggio

Per garantire l'accesso esclusivo al bus occorre arbitrare i tentativi di accesso a questo, in modo tale che un solo contendente alla volta utilizzi la struttura di interconnessione in scrittura. Per quanto riguarda invece le operazioni di lettura, non è necessario alcun arbitraggio dato che, come noto, la lettura contemporanea da parte di più dispositivi non altera il contenuto del bus.

I meccanismi di arbitraggio finora «inventati» possono facilmente essere suddivisi in due categorie: centralizzati e decentralizzati. Nel primo caso esiste un vero e proprio componente, separato dalle unità collegate al bus, che decide chi, in caso di conflitto, avrà la

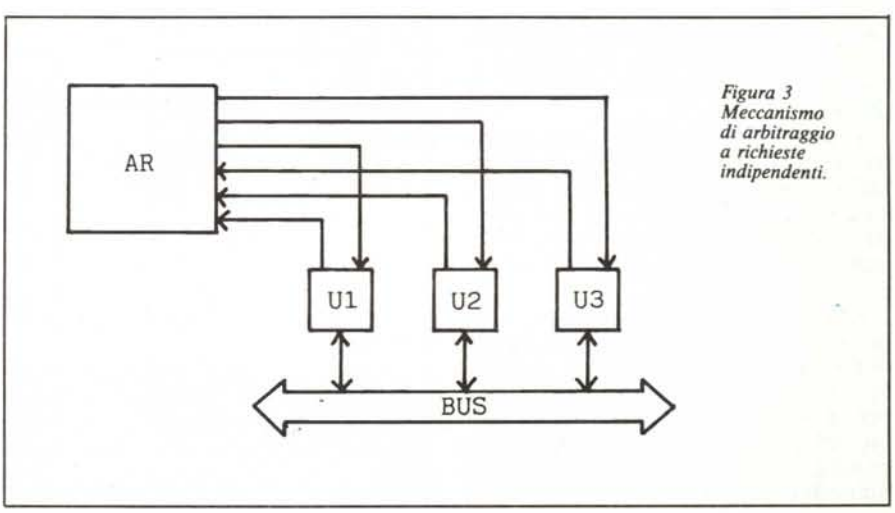

Figura 3 Meccanismo di arbitraggio a richieste indipendenti.

meglio; nel secondo caso, le unità in un certo senso intelligenti saranno in grado di cavarsela da sole senza aiuti esterni né fare pasticci sul bus. Cominciamo coi meccanismi centralizzati.

In figura 3 è mostrato un primo schema di bus arbitrato: pur essendo dei più costosi a realizzarsi, appare come la soluzione più banale. Ogni unità, semplicemente, prima di accedere al bus «chiede il permesso» all'arbitro che, se non ha nulla in contrario, concederà l'utilizzo. Le comunicazioni unità-arbitro, come visibile sempre in figura 3, avvengono tramite collegamenti dedicati nei due versi (si notano infatti due freccie per ogni unità). Se due o più unità chiedono contemporaneamente all'arbitro di utilizzare il bus, riceveranno il consenso a turno secondo una disciplina statica o dinamica a scelta di chi progetterà l'arbitro vero e proprio. Ad esempio, se alcune unità sono più importanti di altre, si può progettare l'arbitro in modo che, in caso di conflitto, dia precedenza a tali dispositivi. Oppure si può scegliere una disciplina pseudo casuale in cui, ad esempio, se in un precedente conflitto ha «vinto» A su B e C la prossima volta si prediligerà B e la prossima volta ancora C. Da sottolineare che in tutti i casi, se il richiedente è unico e il bus è libero, l'arbitro concede d'ufficio la «vittoria» all'unico contendente.

In figura 4 è mostrato uno schema di arbitraggio ben più semplice del precedente ma, come vedremo, con alcune limitazioni. Questa volta il collegamento unità-arbitro non è dedicato, ma ripartito: un altro piccolo bus di servizio. Questo è composto da due sole linee, Richiesta e Occupato, ambedue dalle unità all'arbitro e non viceversa. L'arbitro, dal canto suo, dispone di una linea di uscita Disp che, collegata alla prima unità, da questa alla seconda e così via, permette di far propagare le sue risposte.

Vediamo il funzionamento: quando una unità intende utilizzare il bus in scrittura, manda la sua richiesta sull'omonima linea del bus di servizio. Se più di una unità fa lo stesso non importa: le due richieste si sommeranno (restando però indistinguibili) e arriveranno all'arbitro come «una o più unità ha chiesto il bus». Immaginiamo che al momento attuale il bus sia libero (l'arbitro, come vedremo, lo sa): è sufficiente per l'arbitro mandare il segnale di disponibilità alla prima unità la quale, se non aveva richiesto l'uso, propaga tale segnale all'unità successiva. Se invece l'unità che riceve il segnale di disponibilità è effettivamente interessata all'operazione di scrittura, blocca la propagazione e accede al bus segnalando all'arbitro, con la linea Occupato che da quel momento

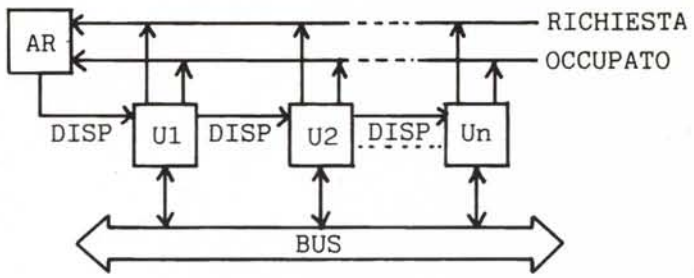

Figura 4 - Meccanismo di arbitraggio Daisy Chaining.

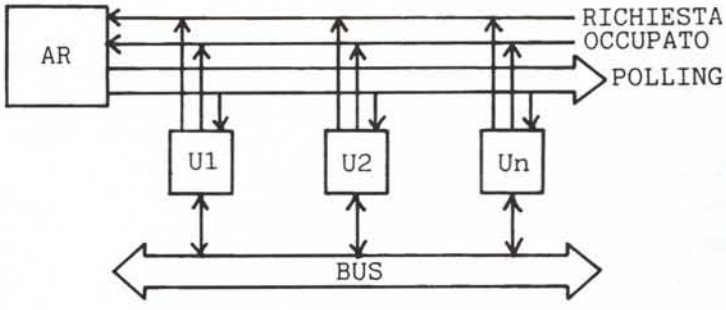

Figura 5 - Meccanismo di arbitraggio Polling.

questo non è più libero. Terminata la sua operazione di scrittura, resettando la stessa linea di prima comunica che il bus è di nuovo disponibile. Il grosso svantaggio di questa architettura è che, come abbiamo visto, le unità più vicine all'arbitro avranno sempre la meglio su quelle più lontane, col ben noto rischio di attesa infinita da parte dei «più deboli». Sempreché non eravamo interessati proprio a questo...

In figura 5 abbiamo una variante del precedente schema, con la quale è possibile evitare l'inconveniente di cui sopra. Al posto della linea di disponibilità nuda e cruda utilizziamo un altro bus di servizio col quale l'arbitro esegue un vero e proprio appello per individuare chi ha fatto la richiesta. Agendo infatti sulle linee di polling (vedi sempre figura 5) invia la disponibilità nell'ordine che vuole, magari riprendendo dalla unità successiva a quella che ha appena liberato il bus.

Tanto la seconda che la terza soluzione hanno l'inconveniente che un'unità, posto ad esempio che il bus è libero, prima di ricevere il consenso, vuoi per la propagazione, vuoi per l'appello, finisce per attendere un tempo ben più lungo del «botta e risposta» dello schema di figura 3. A fronte però di questo svantaggio occorre segnalare che per quanto riguarda l'espandibilità del sistema, ovvero la possibilità di inserire nuovi elementi utilizzanti lo stesso bus, questa è massima nello schema di figura 4 (basta inserire la nuova unità in qualsiasi punto della catena), media nello schema ad appello (polling, in inglese) in quanto ci devono essere sufficienti linee sul bus di polling per ospitare nuovi arrivi e in più occorre comunicare in qualche modo all'arbitro tali variazioni; pressocché nulla nello schema a richieste indipendenti. In questo caso, infatti, per aggiungere una nuova unità occorre cambiare interamente l'arbitro o sovradimensionarlo anzitempo (ma quanto?) in previsione di future espansioni.

Arbitri decentralizzati

I meccanismi di arbitraggio decentralizzati costano poco, hanno espandibilità totale, funzionano egregiamente, ma... procediamo con ordine.

Presenteremo due tipi di arbitri decentralizzati, uno deterministico, l'altro non deterministico. Diciamo che in questo secondo caso, la riuscita o no di un accesso al bus è un po' legata al caso. Sembra una barzelletta ma funziona: ne riparleremo dopo. Per prima cosa ricordiamo che, in uno schema di arbitraggio decentralizzato non esiste un particolare apparato preposto a consentire gli accessi esclusivi, ma le stesse unità aumentate della logica necessaria, riescono a spartirsi amichevolmente il bus senza "fare confusione".

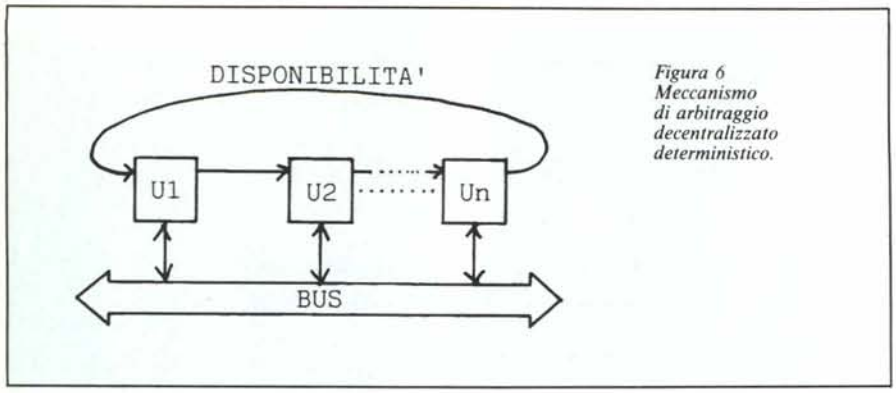

Figura 6
Meccanismo di arbitraggio decentralizzato deterministico.

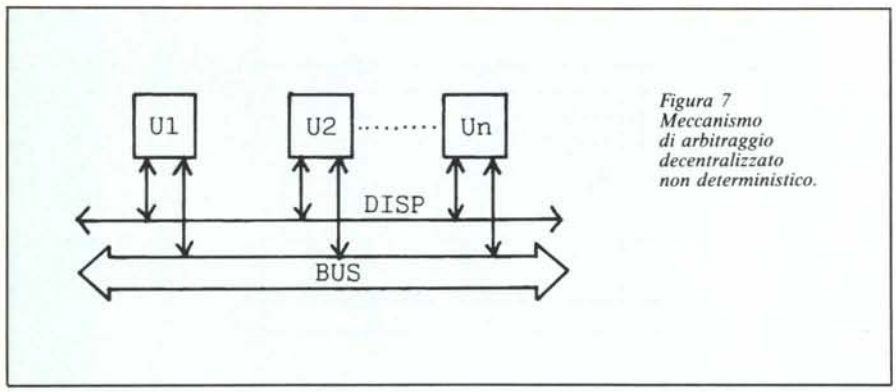

Figura 7
Meccanismo di arbitraggio decentralizzato non deterministico.

In figura 6 troviamo il primo esempio, deterministico. Le unità oltre ad essere collegate al bus, tramite una linea di disponibilità fanno circolare un omonimo segnale con disciplina circolare: U1 lo passa a U2, U2 a U3 e così via fino all'ultimo della catena (nella figura Un) che lo passa nuovamente a U1. Questo accade quando il bus è libero. Se ad un certo momento una unità vuole scrivere sul bus, aspetta di ricevere la disponibilità dall'unità che la precede e sospendendo la propagazione del segnale può tranquillamente accedere alla struttura di interconnessione comune. Terminato l'utilizzo sarà sufficiente che l'unità in questione faccia ripartire la disponibilità affinché tutto torni come prima. Lo svantaggio è quello tipico delle discipline circolari in cui una unità se necessita del bus subito dopo aver ceduto la disponibilità deve attendere un intero «giro». È deterministico dato che chiunque abbia necessità del bus prima o poi verrà accontentato. Cosa che non è detto che succeda col prossimo schema.

Arbitraggio non deterministico

Il meccanismo di arbitraggio certamente più simpatico è quello non deterministico, forse perché dà al funzionamento un tocco di elasticità mentale comune al comportamento umano e poco comune a tutti i rigidi schemi di funzionamento computerecci.

Per farla breve una unità che intende utilizzare il bus, deve soltanto accertare (vedi figura 7) che la linea di disponibilità sia resettata, ovvero che il bus sia libero. Fatto questo setta tale linea e scrive il suo messaggio sul bus senza però settare la corrispondente linea di Ready con la quale autorizzerebbe il destinatario a prelevare il messaggio. Può infatti succedere che nello stesso istante due o più unità, accertato che la linea di Disponibilità è resettata, scrivino sulla struttura di interconnessione provocando la somma dei messaggi che non è significativa per nessuno dei destinatari. Come fa, dunque, una unità a stabilire che sia l'unica a scrivere? Semplice: prima di dare il Ready al destinatario prova a rileggere il suo messaggio per vedere se è uguale a quello che aveva scritto. Se è tale, vuol dire che gli è andata bene e può dare il Ready, altrimenti occorre liberare il bus (operazione che compieranno tutti coloro che sono entrati in collisione) e riprovare dopo un po'. Il buon funzionamento dipende molto da questo «po'», diverso per ogni unità e calcolato accuratamente (incrociando possibilmente anche le dita), ma soprattutto occorre che le unità non usino continuamente il bus altrimenti queste passeranno il tempo a fare tentativi più che ad avanzare con l'elaborazione.

E questo in informatica non è cosa buona e giusta...

MC

Windows.
PC con vista sul futuro.

go al Presentation Manager che userete per comunicare con OS/2.

Direttamente da un'applicazione all'altra.

Avviate un'applicazione; poi un'altra e un'altra ancora. Ritornate indietro per inserire dei dati e riprendete il vostro ultimo programma nel punto esatto dove lo avevate lasciato.

WINDOWS può gestire più programmi contemporaneamente, mantenendoli tutti attivi in memoria, superando agevolmente la barriera dei 640 K del vostro PC.

Con WINDOWS è molto facile il passaggio di informazioni da un programma all'altro, estrarre dati da un programma, combinarli, formattarli e integrarli con informazioni di altri programmi. Così potete anche scrivere applicazioni che dialogano fra loro per la richiesta e il passaggio di dati senza il vostro intervento.

Il vostro "integrato" personalizzato: un ponte sul futuro.

WINDOWS, il potente strumento per i programmi di oggi è una solida base per la nuova generazione di applicazioni in ambiente WINDOWS in via di sviluppo. Write e Paint, per esempio, vi offrono un ricco ambiente operativo che vi permette di unire grafici e testo e di usare caratteri di differenti stili e dimensioni per preparare documenti professionali di alta efficacia visiva.

Tutto questo, e molto altro ancora, unito alla capacità di multitaskig di WINDOWS, vi permette di offrire soluzioni software con potenza e flessibilità sinora impossibili. Un altro successo Microsoft per il vostro successo.

Tante novità.

E fra i tanti successi Microsoft, potete scegliere i nuovi programmi per il calcolo più veloce e potente (Multiplan 3); per la grafica (Chart 2 e Chart 3); per l'archiviazione dati "costruttiva" (RBase System); i linguaggi più evoluti e, per chi vuole risolvere ogni problema con un solo prodotto, il nuovissimo integrato Microsoft Works.

Per maggiori informazioni scrivete o telefonate a:
Microsoft S.p.A.
20093 Cologno Monzese (MI)
Via Michelangelo, 1 - Tel. 02/2549741

GruppoEthos

Microsoft®

Il software del tuo successo.