Algoritmi e architetture dei sottosistemi grafici

Il frame buffer

di Giuseppe Cardinale Ciccotti

Tutti i moderni dispositivi grafici il cui output è indirizzato al monitor, sono basati sul medesimo tipo di architettura che va sotto il nome di «frame buffer». La relativa semplicità costruttiva e la facilità di programmazione hanno creato i presupposti per una crescente diffusione del frame buffer; l'abbattimento dei costi delle RAM dinamiche, principale componente di tali dispositivi, ne ha di fatto permesso l'implementazione anche in tutte quelle situazioni che richiedevano grosse quantità di memoria. La pressoché totale diffusione del frame buffer su tutte le categorie di sistemi grafici, dal personal al terminale grafico di un mainframe, ha poi spinto molti costruttori di componenti a realizzare dispositivi appositamente progettati che consentissero di superare alcuni dei problemi intrinseci all'architettura frame buffer. Anzi il mercato di questi dispositivi è così florido che esistono aziende, come per esempio la Brooktree, specializzate nella progettazione e realizzazione di tali componenti

L'implementazione del frame buffer

Con frame buffer si intende una zona di memoria destinata a contenere un blocco di informazioni, l'ordine relativo delle quali è significativo. Questa definizione prescinde dal tipo di informazioni coinvolte e dal tipo di supporto adoperato per la memoria. Nel nostro caso le informazioni sono di tipo grafico nel senso che ad ogni bit di informazione corrisponderà un punto sul video, un pixel, è naturale perciò pensare il frame buffer come una zona di memoria rettangolare le cui dimensioni verticali ed orizzontali coincidono con quelle in pixel del monitor. Il supporto di memoria può essere di varia natura, nessuno vieta di realizzare un frame buffer su disco o su qualsiasi altro supporto. Tuttavia per ragioni di interazione e di frequenza di refresh dell'eventuale dispositivo raster-scan abbinato, si preferisce implementare il frame buffer mediante memorie a semiconduttore. Anche in questo caso sono possibili diverse alternative: è stata proposta, per esempio, l'implementazione mediante «shift register», vedi figura 1.

L'implementazione tramite shift register prevede perciò un numero di registri a scorrimento pari al numero di pixel in orizzontale. Ogni registro sarà di lunghezza pari al numero di pixel in verticale. In questo modo ogni shift register contribuisce ad un solo pixel per riga. Quando è necessario estrarre i valori dei pixel relativi ad una certa riga per formare l'immagine sul monitor, basta eseguire lo shift di un posto su tutti gli shift register per estrarre le informazioni relative ad un'intera riga. Tuttavia questa operazione non deve essere distruttiva perciò è necessario che i bit relativi ad ogni pixel «rientrino» in ingresso allo stesso shift register. Naturalmente l'interazione è assai abbastanza scarsa perché ogni qualvolta si richiede una scrittura nel frame buffer, è necessario far scorrere tutte le righe del frame buffer, fino a che la riga a cui appartiene il pixel di cui voglio cambiare valore non sia presente nell'ultima posizione nel frame buffer. Le operazioni di scrittura sono perciò assai lente. L'esigenza di poter effettuare operazioni di scrittura in tempi brevi, ha portato i progettisti ad utilizzare RAM dinamiche per memorizzare i dati relativi ai pixel. Con questo tipo di dispositivi, le operazioni di scrittura hanno un tempo costante, permettendo di eseguire operazioni grafiche, come per esempio il riempimento di aree, in tempi predeterminabili a priori. La velocità di scrittura è un parametro fondamentale perché determina il campo applicativo del sistema grafico. In un tipico caso, quello dell'animazione in tempo reale, se il raster ha risoluzione 1280x1024 pixel e desiderando un movimento fluido, la frequenza di rigenerazione della scena è fissata a 30 quadri al secondo, si dispone di soli (1280x1024)/8*(1/30) = 203.45 ns per scrivere ciascun byte del frame buffer. I dati relativi ai pixel nei banchi di memoria del frame buffer, saranno quindi letti con la frequenza verticale del monitor su cui si forma l'immagine grafica; quest'ultima frequenza è fissata strettamente, tale operazione, detta «video refresh», la priorità massima rispetto a qualsiasi altra operazione sul frame buffer stesso. Naturalmente questo problema può essere risolto generando un interrupt ogni 1/(freg. vert.) secondi, tuttavia come si può ben intuire questa interruzione comporta un rallentamento di tutte le operazioni grafiche.

Frame buffer a colori

Fino ad ora abbiamo considerato sempre frame buffer in cui ad ogni pixel corrisponde un solo bit di informazione. È quasi superfluo dire che in questo modo il pixel potrà rispecchiare soltanto questi due stati, sarà perciò acceso o spento e l'immagine risultante risulterà monocromatica. Un dispositivo raster a colori può

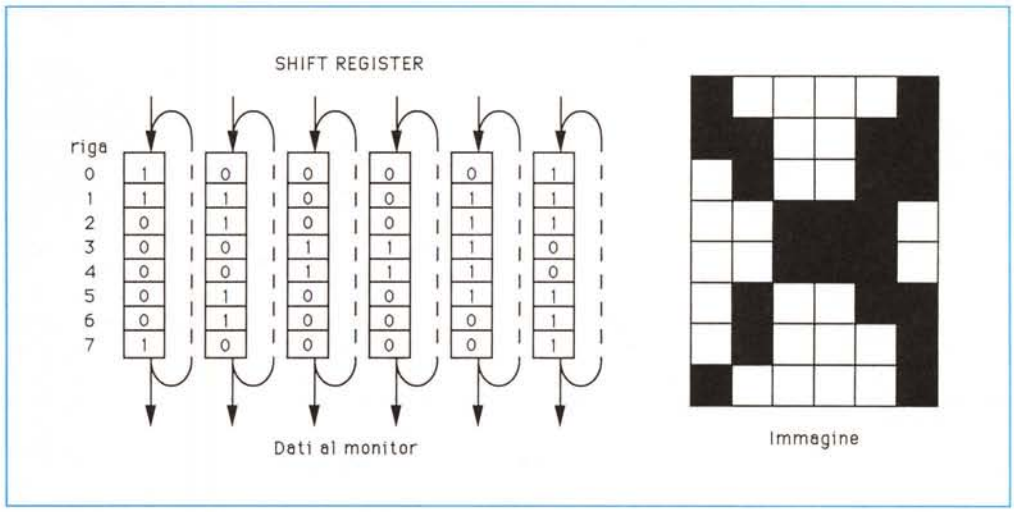


Figura 1
Schema di frame buffer a shift register.

essere pilotato con lo stesso criterio, a patto di fornire informazioni sufficienti a discriminare i colori. Basterà perciò predisporre più bit per ciascun pixel in modo da poter rappresentare più numeri ad ognuno dei quali si farà corrispondere un colore. Per esempio volendo immagini grafiche a 8 colori, ogni pixel sarà individuato da 3 bit e si assumerà ad esempio che alla configurazione 000 è associato il nero, quando i tre bit sono 001 avremo il blu, per 010 si otterrà il verde e così via fino a definire 8 colori scelti a priori dai progettisti del frame buffer. Volendo un numero maggiore di colori è necessario ampliare ancora il numero di pixel in ragione del logaritmo in base 2 del numero dei colori (in seguito si ometterà che il logaritmo è in base 2). Si comprende che volendo rendere la caratteristica del colore indipendente per ogni pixel è necessario disporre di log(n.colori) bit per pixel e perciò l'occupazione di memoria del frame buffer sarà log (n.colori) volte più grande rispetto al caso monocromatico. Una maggiore occupazione di memoria influisce sui tempi di accesso, perché negli stessi intervalli sarà necessario accedere a più locazioni. Riferendoci ad un frame buffer con risoluzione 1280x1024 a 256 colori, valutiamo il tempo di accesso al pixel durante il video refresh a frequenza di 60 Hz: il tempo di pixel sarà pari a (1/60)/(1280x1024xlog(256))=1.5 ns a pixel cioè 25 ns a word! Si vede come con i dispositivi attuali non è possibile ottenere queste velocità se non prendendo contromisure adeguate.

Uno schema generale di architettura

Per risolvere il problema dei tempi di accesso e di decidere una architettura particolare per il frame buffer è necessario puntualizzare quanto costa in termini di cicli macchina ciascun accesso al frame buffer. È concettualmente più intuitivo considerare il frame buffer come una zona di memoria indirizzata da due coordinate, una orizzontale x e una verticale y. Tuttavia la RAM del frame buffer deve essere indirizzata linearmente, perciò è necessario convertire la rappresentazione bidimensionale x,y in rappresentazione a lista lineare. Assumendo che l'indirizzo di partenza della zona di memoria è diverso da zero la formula di conversione è la seguente:

Indirizzo (x,y) = (xmax-xmin) * (y-ymin) + (x-xmin) + indirizzo base frame buffer
xmax,xmin = coordinate orizzontali minime e massime del frame buffer in pixel
ymin = coordinate verticali minime del frame buffer in pixel

Bisogna tener presente che l'origine degli assi x,y è posta nell'angolo in alto a sinistra del frame buffer e che per un dato frame buffer xmax, xmin e ymin sono costanti fissate. Quindi il calcolo dell'indirizzo può essere semplificato riscrivendo l'equazione in termini dei tali costanti:

Indirizzo (x,y) = K1 + K2*y+x
dove K1 = indirizzo base frame buffer-K2*ymin-xmin
K2 = xmax-xmin

quindi il calcolo dell'indirizzo del pixel richiede soltanto due addizioni e una moltiplicazione, in particolare la moltiplicazione può essere eliminata quando si possa indirizzare i pixel in modo incrementale:

Indirizzo (x+1,y) = K1+K2*y+x+1 = Indirizzo (x,y)+1
Indirizzo (x,y+1) = K1+K2(y+1)+x = Indirizzo (x,y)+K2
Indirizzo (x+1,y+1) = Indirizzo (x,y)+K2+1

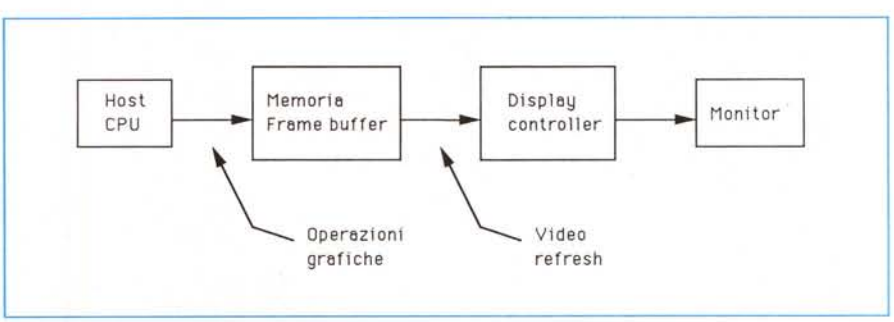


Figura 2 - Schema funzionale di sottosistema grafico a frame buffer. Il sistema si compone di più elementi e due canali logici.

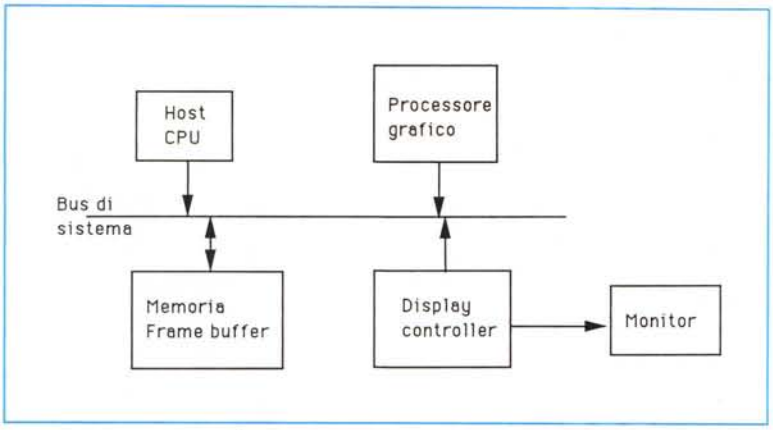


Figura 3 - Schema a blocchi di architettura frame buffer con processore grafico periferico. Il bus è condiviso e ci sono collisioni per l'accesso.

Naturalmente ai simboli di somma possono sostituirsi quelli di sottrazione.

Da queste considerazioni si può capire perché il frame buffer ha uno schema funzionale a blocchi come quello rappresentato in figura 2, bisogna notare come il frame buffer ha funzionalmente un canale di ingresso ed uno di uscita nettamente separati: in tal modo si riescono a soddisfare i vincoli di tempo imposti da un lato dalla frequenza video e dall'altro dalla necessità di eseguire sulla medesima zona di memoria, operazioni grafiche. Questo tipo di schema implica la presenza di due dispositivi distinti: una unità di processo e un «display controller»; la prima, nelle realizzazioni più semplici, può essere la stessa CPU del sistema di elaborazione di cui il sottosistema grafico fa parte, il secondo è invece un dispositivo assai particolare in quanto ha il compito di interfacciare dispositivi digitali e analogici. È ovvio che il più semplice funzionamento di questo schema ricalca quello di una comune unità periferica di un processore: è la CPU centrale che si occupa sia di gestire le operazioni grafiche che quelle di refresh video, il display controller sarà soltanto un convertitore ADC e avrà un timer per generare gli esatti sincronismi video. Non sarà però in grado di accedere direttamente alla memoria, al massimo genererà un interrupt per la CPU. È chiaro che questo modo di funzionamento non può assicurare prestazioni di rilievo in quanto anche adottando processori a 32 bit delle ultime generazioni non si potrebbero ottenere risoluzioni maggiori di 640x480 con 256 colori, infatti il tempo disponibile per l'accesso ad una doppia word è pari a (1/60)/(640x480x8/32) = 217 ns comparabile con i tempi di lettura di un microprocessore a 32 bit. Al di là di questi semplici considerazioni numeriche, un sistema nel quale la CPU si interrompe ogni 20 ms circa non è proponibile. Conviene allora predisporre un display controller in grado di accedere direttamente alla memoria e quindi in grado di sollevare il processore da questo gravoso compito; il display controller avrà allora anche un generatore di indirizzi per la RAM. La CPU per conto suo, se il resto del sistema non lo richiede potrà essere anche meno potente di quanto si era stabilito, comunque la generazione della immagine video sarà assicurata dal display controller che avrà naturalmente accesso alla RAM con priorità rispetto alla CPU. Tuttavia se si evita di fermare la CPU durante il refresh video ma si interdisce l'eventuale accesso di quest'ultima alla RAM del frame buffer, si ha la possibilità di non fermare la computazione durante il video refresh. È questo lo schema di funzionamento delle schede grafiche di molti personal computer.

Schemi funzionali di architetture evolute

Superato lo scoglio delle interruzioni dovute al video refresh, è naturale tentare di risolvere i problemi legati alla velocità di esecuzione delle operazioni grafiche. Nello schema che abbiamo finora analizzato la CPU centrale dovrà incaricarsi di ogni operazione che l'utente vuole eseguire sul frame buffer, anche la modifica del valore relativo ad un pix-

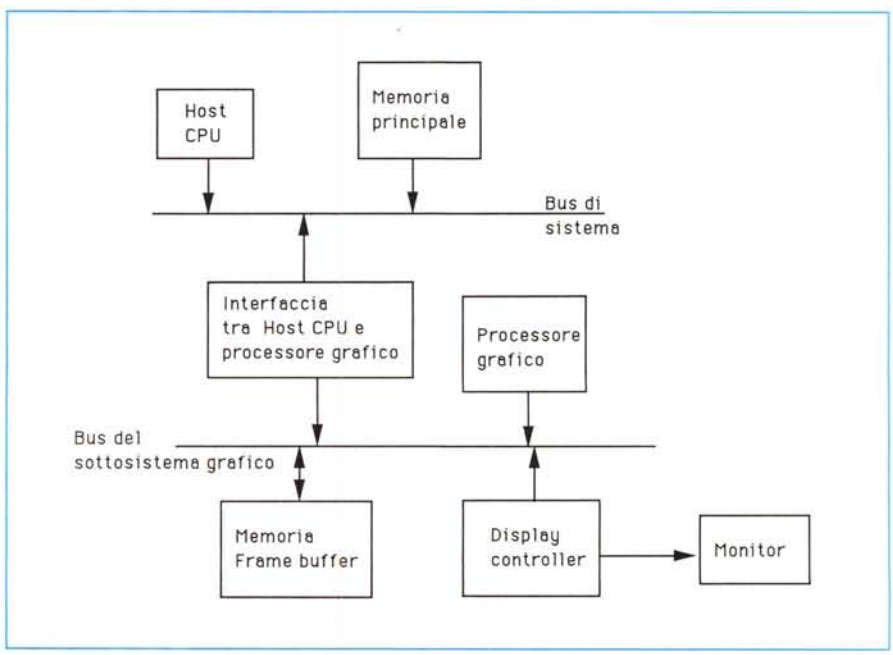
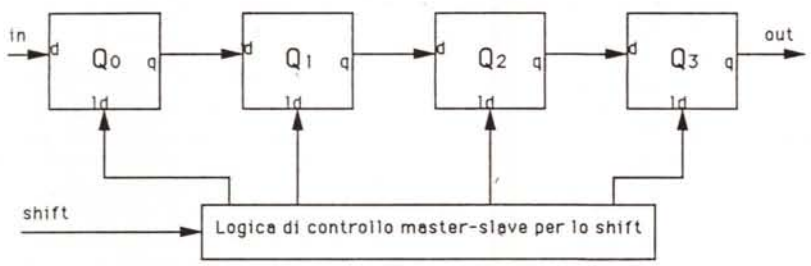


Figura 4 - Schema di architettura di sottosistema grafico a doppio bus. La CPU del sistema host può accedere alla memoria del frame buffer solo indirettamente tramite il processore grafico a cui è collegato mediante un'interfaccia dedicata.

Lo shift register

Uno shift register, o registro a scorrimento, può essere rappresentato come una fila di flip-flop, in cui l'uscita di ognuno è connessa all'ingresso del successivo. Agendo sul comando di «load» di ciascun singolo flip-flop si potrà fare in modo di caricare il dato presente all'ingresso che è ovviamente quello disponibile sull'uscita del precedente, in tal modo il bit «scorre» da un flip-flop al successivo. Si potranno perciò caricare tanti bit nel registro, quanti sono il numero dei flip-flop nella fila, ma soltanto inserendoli all'ingresso del primo che non avendo predecessori è libero; allo stesso modo i bit potranno essere prelevati solo dall'uscita dell'ultimo flip-flop che non ha successori. Lo shift register ha perciò un solo ingresso e una sola uscita; nel caso in cui il registro fosse «pieno», avessimo cioè già immesso un numero di bit pari al numero dei flip-flop presenti e volessimo inserire un ulteriore bit, sarebbe necessario fare «posto» a quest'ultimo. Caratteristica dello shift register è che è il bit nell'ultimo flip-flop della fila ad essere «spinto fuori». A partire dal penultimo perciò tutti i bit sono copiati sul flip-flop successivo, in sequenza, e alla fine dopo che il primo bit è passato nel secondo flip-flop, il nuovo bit è caricato. Si evince che ogni qualvolta si scrive un bit, è disponibile sul piedino di uscita il bit caricato n-1 scritture prima, n è pari al numero di flip-flop dello shift register. Naturalmente la scrittura in successione dei flip-flop, trasparente all'esterno, è governata da una logica interna al dispositivo.



Schema semplificato di shift register a 4 flip-flop. Quando il comando shift va ad 1, i bit scorrono nel flip-flop successivo e sul primo è caricato il bit presente sull'ingresso in.

el sarà a carico della CPU. Nel paragrafo precedente abbiamo valutato come sia comodo indirizzare il singolo pixel tramite le sue coordinate x,y, perciò l'accesso al frame buffer non sarà in genere paragonabile ad un semplice accesso in memoria ma ci sarà un overhead dovuto al calcolo dell'indirizzo fisico; sarà perciò necessario eseguire una routine, che in genere sarà disponibile nella libreria del sistema, altrimenti occorrerà programmarsela. La CPU centrale è in genere un processore non specializzato e perciò non possiede particolari caratteristiche che permettano di velocizzare l'esecuzione di questa routine, l'ideale sarebbe poter disporre di un'istruzione a basso livello che esegua la scrittura del pixel direttamente e accetti come argomenti x,y e il colore del pixel. I costruttori di componenti hanno interpretato questa esigenza, progettando e realizzando dei processori specializzati nelle operazioni grafiche, i primi di questi erano delle versioni particolari di normali CPU a cui era stato riscritto il microprogramma, modificando così ad hoc il set delle istruzioni. Naturalmente l'evoluzione di questi dispositivi ha portato ad una specializzazione sempre maggiore tanto che i «Graphic Processor» formano oggi una categoria a parte e implementano, spesso direttamente in hardware, operazioni assai complesse, avremo modo in seguito di parlare di questi processori. Inserendo un processore grafico in un sistema dove è già presente un altro processore, occorre trovare il modo di farli colloquiare. In particolare si dovrà stabilire un protocollo di comunicazione tra i due, decidere se il processore grafico è master o slave rispetto alla CPU centrale, nel senso che la comunicazione dei dati avvenga a richiesta della CPU o del processore grafico e in particolare se la memoria deve ancora essere vista dal processore centrale come parte integrante del suo spazio di indirizzamento o in alternativa predisporre due bus separati uno per la memoria centrale ed uno per la memoria del frame buffer, potete vedere gli schemi che illustrano tali architetture in figura 3 e 4. Nel primo schema il fatto di avere due processori che condividono il bus è sconveniente rispetto alle prestazioni globali perché ci saranno contese per il possesso di tale risorsa. Il secondo schema prevede che l'accesso alla memoria del frame buffer sia consentito soltanto al processore grafico, in tal modo il sottosistema grafico può essere ottimizzato per rispettare anche i vincoli più stringenti. Questo secondo schema consente inoltre di poter progettare il sottosistema grafico in maniera quasi indipendente dal sistema host in quanto è sufficiente modificare l'interfaccia verso il bus nel caso in cui si debba adattare il sottosistema grafico ad un altro host. Questo schema è molto seguito e visto che i messaggi tra l'host e il sottosistema grafico sono brevi e non frequentissimi, specialmente se il processore grafico è in grado di eseguire da microprogramma o da programmi di libreria operazioni complesse, alcune aziende propongono frame buffer dotati di interfacce standard parallele, seriali o SCSI.

Conclusioni

L'architettura frame buffer ci accompagnerà durante il nostro excursus nel mondo dei sottosistemi grafici, faremo perciò spesso riferimento ad essa. In questo articolo abbiamo analizzato degli schemi abbastanza generali di architetture, seguendo grossomodo l'evoluzione e la nascita dei sistemi grafici. Nei prossimi appuntamenti analizzeremo in dettaglio alcune architetture ad alte prestazioni, tra cui l'architettura bit-plane; conosceremo infine alcuni nuovi componenti come le videoRAM e i color-palette. MS

Bibliografia

David F. Rogers «Procedural elements for computer graphics» McGraw-Hill 1985.