Un'interfaccia avanzata per USB

L'architettura ibrida di ARC International consente di ridurre il time-to-market nello sviluppo delle connessioni USB veloci

Peter Wells, Chris Belanger
ARC International

L'Universal Serial Bus prospera. Pensato per ridurre la molteplicità dei cavi di connessione che attorniano il PC, questo bus seriale è così semplice nella sua concezione da essere diventato oggi uno standard di riferimento per tutti i collegamenti fra i dispositivi digitali. Grazie all'aumento della banda di circa 40 volte introdotto dalle specifiche USB 2.0 e alle norme aggiuntive relative alla connettività degli apparecchi anche non PC e di tipo On-the-Go (OTG), le interfacce hardware USB High-Speed e OTG si stanno imponendo non solo nel tradizionale mercato dei PC, ma anche in quello degli apparecchi portatili e in tutto ciò che ruota attorno a entrambi.
Le nuove varianti sono state implementate insieme ai relativi software nelle specifiche Enhanced Host Controller Interface (EHCI) definite da Intel, NEC, Compaq e altre aziende leader. Queste norme descrivono un set di registri e di strutture dati standard che permettono a un Host USB di comunicare con un altro Host Controller, o con un OTG configurato come Host. Le specifiche, in pratica, raccolgono il meglio di quanto offerto dalle precedenti OHCI e UHCI di USB 1.1 e le sostituiscono totalmente, migliorandole. Il beneficio più importante che offrono è quello di costituire un sistema software integrato e uniforme più piccolo nelle dimensioni, cosa particolarmente importante per le applicazioni OTG, e più leggero da gestire da parte del sistema operativo e dell'hardware di destinazione.

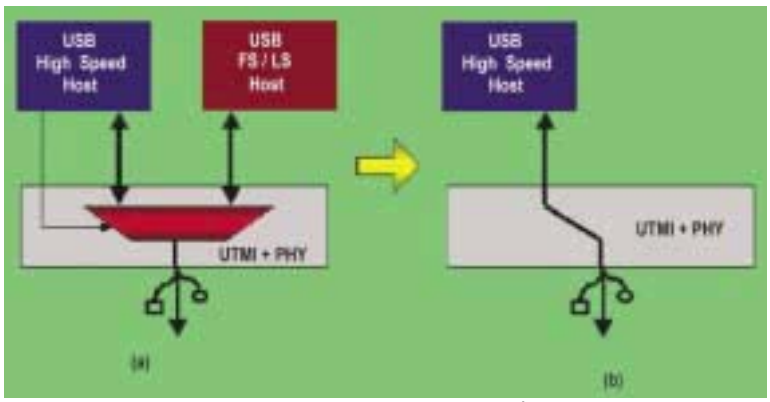

Fig. 1 – Un'interfaccia USB ad alta velocità con (a) e senza (b) controllore intermedio a bassa velocità

Le attuali architetture USB 2.0

Sfortunatamente, l'ottenimento di questi importanti vantaggi dipende dall'esistenza di un gradino intermedio che i produttori di controllori USB sono costretti a prendersi carico. Il passaggio iniziale prevede infatti l'implementazione degli EHCI USB HS (High-Speed) Controller attraverso i controllori intermedi OHCI USB FS/LS (Full-Speed/Low-Speed), come si vede nella figura 1a. Il passo successivo consiste nell'implementare i controllori EHCI come più efficienti Hub 0, in modo da fare a meno dell'impostazione duale (Fig. 1b). Nonostante la prima soluzione sia quella che consente la più ampia compatibilità con il software esistente e le applicazioni FS/LS a bassa velocità, è però anche la meno efficiente a causa di alcuni non trascurabili "effetti collaterali".
Innanzitutto occorrono due controllori USB al posto di uno. Un controllore, con i registri compatibili con le specifiche EHCI e la capacità di gestire le strutture di memoria EHCI, deve occuparsi di tutto il traffico ad alta velocità (HS) e del traffico a bassa velocità entrante (FS o LS). Un secondo controllore totalmente separato, con i registri compatibili con le specifiche OHCI e la capacità di gestire le strutture di memoria OHCI, deve occuparsi del solo traffico a bassa velocità (FS o LS). Ciò significa che aumentano sia lo spazio occupato sul silicio, sia le necessità d'integrazione hardware e software.
Inoltre, la complessità del livello fisico USB 2.0 PHY aumenta per l'inevitabilità di dover utilizzare sia l'interfaccia Serial Interface Engine (SIE) per il controllore di supporto, sia la Universal Transceiver Macrocell Interface (UTMI) per il controllore ad alta velocità. Ciò implica un maggior numero di pin nell'interfaccia fisica e un più oneroso controllo dei dati in transito sul cavo USB. Due stack di controllo USB devono inoltre essere progettati e implementati per funzionare in modo totalmente indipendente l'uno dall'altro. Un esempio è

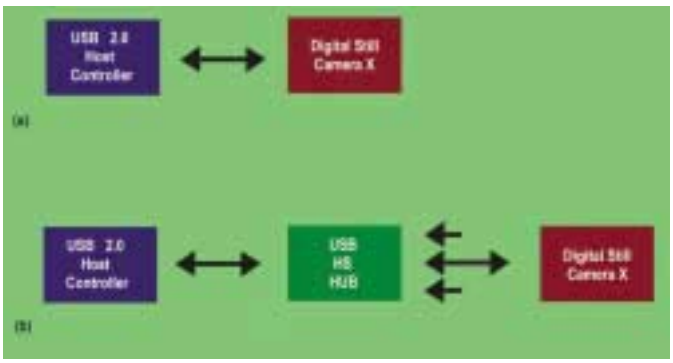

Fig. 2 – La connessione di un dispositivo digitale all'interfaccia USB HS può avvenire direttamente (a), oppure attraverso un Hub (b)

illustrato nella figura 2. Nella 2a si vede un fotogramma di una camera digitale DSC direttamente connesso all'interfaccia USB, che però necessita di un driver OHCI per implementare le funzionalità doppie. Lo stesso fotogramma DSC può essere connesso all'Hub USB ad alta velocità, come si vede nella figura 2b, ma in questo caso avrà bisogno di un apposito driver compatibile con le specifiche EHCI, per poter svolgere le stesse funzioni. Di conseguenza, quest'impostazione obbliga a raddoppiare tutte le risorse necessarie allo sviluppo dell'intero sistema, evidenziando la ridondanza e il "peso" della presenza del controllore OHCI. Al contrario, un'architettura di controllo unificata e basata sulle specifiche EHCI è la soluzione ottima per tutte le applicazioni USB.

Tradurre le transazioni

I controllori USB HS sono la naturale evoluzione delle specifiche EHCI e sono già disponibili sul mercato. Nel capitolo 11 delle normative USB 2.0 è preso come esempio il controllore ARC USB-HS IP prodotto da ARC International, che integra anche complete funzionalità di Transaction Translator. Windows riconosce quest'architettura come un "Supported Host Controller", identificato nel sito Web Microsoft con la frase seguente: "The USB host controller may or may not contain companion controllers. If no companion host controllers are present, Microsoft requires the host silicon expose a self-powered USB 2.0 hub (with a Transaction Translator) . . ."
Il modulo embedded Transaction Translator (TT) deve apparire come Hub 0 nelle strutture dati EHCI e deve occuparsi di convertire il traffico FS e LS in transazioni separate di tipo particolare come prescritto nelle specifiche USB 2.0. Questo approccio consente di fare in modo che un'unica implementazione USB EHCI possa essere in grado di gestire direttamente tutti i dispositivi HS, FS e LS, senza bisogno di ricorrere al software OHCI, al controllore ad esso dedicato e alle ingombranti esigenze di maggiori risorse elettriche, logiche e fisiche.
L'architettura dell'Hub 0 embedded è più leggera, più efficiente e consuma meno, tanto da poter essere utilizzabile anche negli apparecchi portatili. Oltretutto è in grado di offrire prestazioni superiori in termini di banda, adatte alle applicazioni embedded più evolute come, ad esempio, i set-top-box e i gateway residenziali. D'altra parte, mentre i dispositivi Hub 0 embedded hardware sono già disponibili sul mercato, lo stesso non si può dire per il software, che sembra giungervi un po' a rilento. Ciò è dovuto al fatto che le specifiche EHCI non descrivono molto bene le funzioni software degli Hub 0 e, inoltre, solo recentemente sono stati proposti dei controllori embedded con funzionalità TT.
Per usare le funzionalità TT embedded nei sistemi operativi in commercio ci sono due modi: trasferire le specifiche USB supportate dal dispositivo hardware all'interno del sistema operativo prescelto, al posto delle specifiche USB residenti in esso; trasferire l'Hardware Abstraction Layer delle specifiche USB residenti nel sistema operativo in una nuova architettura hardware. Entrambe le strategie richiedono un minimo di sforzo d'ingegnerizzazione.

Implementare ARC USB a bordo di un sistema operativo embedded

L'Host USB ARC International, i dispositivi OTG e i driver specifici sono stati progettati per essere portabili. Innanzitutto, sono stati scritti in ANSI C senza alcuna istruzione che richieda esplicitamente l'intervento del sistema operativo
Inoltre, usano macro che sono raccolte in un file d'intestazione, che può essere modificato per usare istruzioni dedicate prese dalle librerie di qualsiasi RTOS. Ciò consente di

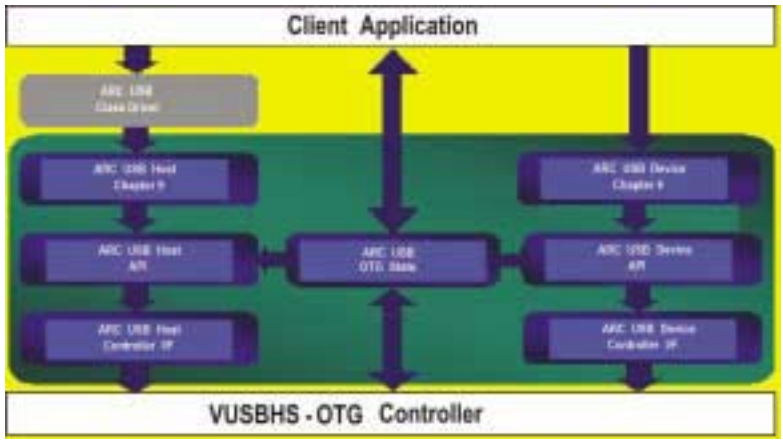

Fig. 3 – L'architettura del software ARC USB OTG

Tabella 1 – Le funzioni macro definite nell'RTOS MQX

#define USB_lock()	_int_disable()
#define USB_unlock()	_int_enable()
#define USB_memalloc(n)	_mem_alloc_system_zero(n)
#define USB_memfree(ptr)	_mem_free(ptr)
#define USB_memzero(ptr,n)	_mem_zero(ptr,n)
#define USB_memcopy(src,dst,n)	_mem_copy(src,dst,n)
#define USB_install_isr(vector, isr, data)	_int_install_isr(vector, isr, data)

trasportare ARC USB su qualsiasi sistema operativo embedded come Linux, VxWorks o Windows CE, per una maggior versatilità d'integrazione nei sistemi. In realtà, questo è importante perché alcuni sistemi operativi non hanno il completo supporto dei protocolli EHCI e OTG. Le funzioni dell'RTOS che possono essere utilizzate dall'interfaccia USB sono solo quelle relative alla gestione delle memorie e degli interrupt. Queste funzioni sono raggruppate nel file predefinito d'intestazione e possono facilmente essere modificate, qualora sia necessario per esigenze di portabilità. A titolo di esempio, usando l'RTOS MQX di ARC queste macro sono definite come nella tabella 1. Se necessario, si possono aggiungere ad esse le istruzioni specifiche proprie del particolare sistema operativo prescelto. Un file d'intestazione tipico dell'MQX è mostrato nella tabella 2. ARC fornisce le funzioni adatte alla maggior parte degli eventi occorrenti nelle comunicazioni USB, così tutte le funzioni specifiche dell'OS possono essere personalizzate a questo livello.

Fra esse, vi sono le mutue esclusioni, i semafori, gli eventi improvvisi, i messaggi e tutte le segnalazioni tipiche del protocollo. Le funzioni applicative possono essere definite come task del sistema operativo in tempo reale e questo implica che nell'RTOS la gestione delle task viene svolta a livello applicativo. Nell'MQX, i registri delle task sono controllati a basso livello e, a questo proposito, ARC fornisce dei codici applicativi di esempio già pronti e direttamente utilizzabili su qualsiasi piattaforma software e qualunque sistema operativo. La figura 3 mostra l'architettura alla base del core ARC USB OTG.

Implementare USB a bordo di un controllore ARC USB HS OTG

Il secondo modo per usare l'hardware ARC USB insieme con un sistema operativo embedded consiste nel trasferire le istruzioni USB proprie del particolare sistema operativo direttamente a bordo dell'hardware ARC. Tuttavia, questo può essere fatto solo al più basso livello del protocollo, lasciando intatti i livelli applicativi più alti. Il primo requisito è che l'RTOS sia in grado di supportare entrambe le specifiche EHCI e OTG, anche se a livello dei soli moduli Host è sufficiente la compatibilità EHCI, supportata comunque da Linux. Ogni implementazione richiede in teoria delle API differenti e quindi questa fase può essere diversa per ogni sistema, ma generalmente le modifiche possono essere raggruppate nelle seguenti sei categorie:

Tabella 2 – Un file d'intestazione tipico di ARC MQX

```
#ifdef __USB_OS_MQX__
   #include "bsp.h"
   #include "fio.h"
   #include "mqx_arc.h"
#else
...
#endif
```

1 - Host Mode Initialization
2 - Controller Reset and Port Control
3 - USB Connection Speed Detection
4 - Support for Embedded TT vs. USB connected TT in HUB
5 - Test Mode Support
6 - Exclusion and Stubbing of OHCI stack

Vediamo ciascuna di esse con, ove possibile, un esempio su Linux.

1 - Inizializzazione della modalità Host

Le specifiche EHCI assumono sempre che tutti i controllori siano Host Controller. In qualsiasi core OTG le funzionalità operative dell'Host sono controllate dal software, ma nell'architettura ARC USB HS OTG il codice può essere modificato solo durante l'inizializzazione dell'Host. Negli USB HS Host, pertanto, non è necessaria alcuna modifica.

2 - Reset del controllore

Ci sono alcune piccole modifiche da implementare affinché ARC USB HS riconosca gli eventi Reset. In Linux ci sono dei temporizzatori software che sono utilizzati per attendere il minimo tempo necessario al reset del sistema prima di riprendere l'elaborazione. Nel core ARC questa funzione viene svolta in forma hardware (così, se i temporizzatori software sono configurati su un tempo di attesa superiore a quello imposto dai timer hardware, non è necessaria alcuna modifica). Qualora si debba configurare il core OTG per le sole funzioni Host, devono essere fatte delle modifiche per inizializzare in modo corretto i registri EHCI Port Status. Queste correzioni adeguano lo stato iniziale della porta USB per il controllo dell'accesso alla porta stessa.

3 - Misura della velocità nella connessione USB

Le specifiche EHCI permettono all'Host di determinare la velocità di un dispositivo connesso campionando lo stato della linea tramite un opportuno registro, durante un certo intervallo di tempo successivo al reset del bus. Il core ARC USB HS ha un registro apposito che può essere utilizzato per questa funzione, in modo totalmente compatibile con le norme EHCI. In alternativa, può anche rilevare la velocità della connessione usando solo i componenti hardware e immagazzinare tali informazioni nella parte libera del registro di Port Status non utilizzata dal software EHCI, in modo tale che tutta la procedura di temporizzazione software prescritta dalle specifiche EHCI possa essere bypassata se necessario.

4 - Supporto delle funzionalità TT
È una modifica un po' laboriosa per l'hardware ARC, che serve a implementare le funzionalità TT a bordo dell'Hub 0, in modo che questo possa gestire le transazioni a livello dei propri registri. In pratica, consiste nell'inserire nella procedura "ehci_hub_control" le funzioni ClearTTBuffer, ResetTT, GetTTState e StopTT. A questo proposito, ARC International ha reso disponibili degli esempi che chiariscono meglio l'operazione.
5 - Supporto delle modalità di test
Le specifiche EHCI descrivono una modalità di test in cui una stringa predefinita di pacchetti può essere ripetuta più volte attraverso la connessione USB. Questa modalità non operativa può essere utilizzata per il debug e il collaudo dei sistemi. ARC ha implementato questa funzione in forma software per non ingombrare il core hardware.
6 - Esclusione e bypass dello stack OHCI
Un importante vantaggio associato all'uso del core ARC USB HS è che l'ingombro totale del codice può essere significativamente ridotto togliendo tutto ciò che riguarda il protocollo OHCI. Con gli opportuni accorgimenti si può fare in modo che né il software né l'hardware abbiano bisogno di alcuna funzione OHCI. Certamente, ci possono essere altre modifiche tipiche di ogni particolare piattaforma software. Esempi di questo tipo sono i registri opzionali PCI prescritti nelle specifiche EHCI e il codice dedicato all'uso di DMA esterne, per quelle architetture nelle quali non è compreso internamente il controllo DMA come nel core ARC. In questi casi occorre intervenire singolarmente con modifiche "ad hoc".

Un compromesso utile

L'architettura più scelta oggi per la gestione delle applicazioni USB 2.0 HS/FS/LS è un compromesso necessario per le esigenze di time-to-market. I difetti presenti nei controllori standard possono essere superati implementando in forma embedded le funzioni di Hub Transaction Translator all'interno del controllore USB, in modo da poter eseguire le funzioni EHCI software e hardware sul traffico USB. Queste architetture potenziate, disponibili presso alcuni produttori leader come Intel, Microsoft e ARC International, sono un ottimo approccio per risolvere i problemi di compatibilità e velocità nelle linee USB.
La soluzione offerta da ARC International consente di ridurre le risorse software, i requisiti hardware e il supporto esterno necessari all'esecuzione delle funzioni USB nei sistemi. Per migrare a questa più efficiente architettura bastano poche operazioni alle quali l'azienda può assistere per rendere ulteriormente più semplice l'implementazione dei core ARC nei sistemi USB HS OTG.

ARC International
www.arc.com