IL FUTURO È NELLE LOGICHE ASINCRONE

Possono risolvere numerosi problemi circuitali nei sistemi ad alta frequenza e soprattutto in quelli di grandi dimensioni

Juergen Pintaske

La tecnologia CMOS è, e rimarrà ancora a lungo, sovrana dello sviluppo dei circuiti a semiconduttore. Tuttavia, nel progetto dei sistemi più complessi, dei sistemi-su-silicio di grandi dimensioni e dei dispositivi integranti più blocchi di proprietà intellettuale, pone alcuni problemi per lo più relativi all'ottimizzazione dei segnali di temporizzazione, al contenimento del consumo di potenza ed alla scelta dei tool per il progetto, la verifica ed il test dei circuiti. Una delle maggiori cause di ciò è che comunemente l'intero circuito è disegnato con una singola sincronizzazione delle temporizzazioni. Invece, si possono risolvere questi problemi e migliorare sensibilmente le prestazioni dei circuiti optando per una configurazione asincrona delle temporizzazioni sulle logiche.

La recente evoluzione delle tecnologie di progetto dei system-on-chip, infatti, ha portato all'integrazione di un numero sempre maggiore di funzioni in aree di silicio sempre più piccole. Questo ha come diretta conseguenza l'aumento incrementale del numero di porte logiche e delle difficoltà nell'ottimizzare i segnali di temporizzazione. Di conseguenza si dilatano i tempi di sviluppo, allungando il parametro divenuto oggi più che mai strategico del time-to-market, ovvero la capacità di rendere disponibile un prodotto nel più breve tempo possibile dopo l'inizio del suo progetto. Un modo per accorciare i tempi di sviluppo è quello di utilizzare nel disegno del sistema più blocchi di proprietà intellettuale (IP), scegliendoli dall'ampia offerta presente sul mercato. In questo modo si riducono i tempi, ma si aggiunge il rischio di non conoscere completamente le funzionalità dei blocchi IP prescelti e di imbattersi in errori di progetto che possono anche rivelarsi fastidiosi. Molti integrati ed ASIC utilizzati nelle applicazioni portatili, per esempio, presentano oggi il tipico dilemma costituito dal fatto che, sebbene la corrente circolante dev'essere resa minima possibile per fare in modo che le batterie durino più a lungo possibile, purtroppo l'assorbimento di corrente e quindi il consumo di potenza crescono linearmente con l'aumento della banda di frequenza. Oltretutto, la tensione offerta dalle batterie non è mai perfettamente stabile e, pertanto, occorre sempre consumare una certa percentuale di energia per regolarla e quest'operazione, in genere, causa sempre una significativa dissipazione. Senza contare che elevate frequenze di clock e lunghi circuiti per i segnali di temporizzazione comportano un alto rischio di emissioni ed interferenze, qualora alcune linee mostrino di comportarsi da antenne. Certamente è possibile togliere l'alimentazione alle aree del sistema di volta in volta non attive, ma in tal caso si gonfiano ulteriormente le difficoltà nel progetto circuitale. Si tratta, comunque, di problemi comuni a numerosi ambiti applicativi, per i quali non vi è altra soluzione se non quella di tenere più basso possibile il livello della tensione di alimentazione. Per di più, i tool di pro-

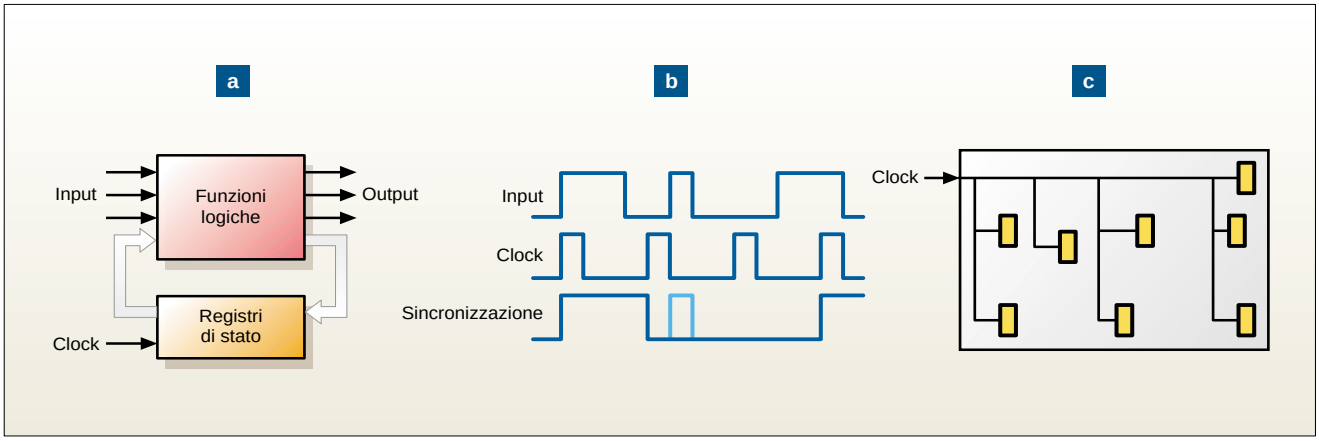

Fig. 1a - In una macchina a stati FSM, Finite State Machine, i registri offrono in ogni momento la configurazione istantanea del sistema

Fig. 1b - La sincronizzazione degli ingressi pone sempre alcuni problemi

Fig. 1c - La rete di distribuzione dei segnali di temporizzazione in un sistema reale può essere molto complessa

Tabella 1 - Gli aspetti fondamentali dell'implementazione asincrona delle logiche

Aspetto	Descrizione
Trasferimento dati basato sul livello dei segnali	0 ed 1 sono come nelle logiche convenzionali. Se la commutazione fra i due livelli è troppo lenta, si manifesta uno stato intermedio definito come "sconosciuto", o "unknow".
Trasferimento dati basato sulla commutazione dei segnali	Il segnale è accettato solo dopo ogni commutazione. Il livello ed i transitori dei segnali sono meno importanti, giacché nel caso peggiore causano solo dei ritardi.
Pacchetti di dati	I pacchetti di dati ed i gruppi di segnali sono accompagnati da una linea di clock addizionale ed indipendente.
Trasferimento segnali Dual-Rail	I livello alto e basso di un segnale sono trasportati da due linee separate.
Indipendenza della velocità	Il sistema elabora l'informazione logica indipendentemente dalla velocità delle porte che può variare se cambiano la temperatura o la tensione di alimentazione.
Insensibilità ai ritardi	Completa indipendenza da qualsiasi ritardo.

getto, verifica e test mostrano di essere talvolta in difficoltà nel gestire i grandi circuiti organizzati con una singola sincronizzazione dei segnali di temporizzazione e soprattutto al momento di eseguire la sintesi e la simulazione.

Le logiche asincrone

Non sono certo una novità, dato che si conoscono da più di vent'anni, ma gli ingegneri hanno sempre esitato ad utilizzarle, preferendo per mille motivi le logiche sincrone e questo spiega perché i tool in grado di supportare le logiche asincrone scarseggino sul mercato. È anche vero che se tutti i segnali presenti in un chip commutano negli stessi istanti la verifica del sistema è più semplice, poiché basta controllare che ad ogni commutazione il ritardo delle uscite dalle logiche combinatorie rimanga inferiore dell'intervallo di tempo di attesa prima del clock successivo. In un sistema con segnali asincroni la verifica è più complessa perché occorre conoscere molto meglio le caratteristiche dei segnali di temporizzazione. Tuttavia le logiche sincrone e le asincrone non si escludono, anzi, in linea di principio l'implementazione sincrona può essere considerata un caso particolare della più generale implementazione asincrona.

Ma come lavora un sistema sincrono e come può essere trasformato in un sistema asincrono? Innanzi tutto, in un sistema sincrono tutti gli ingressi e tutte le uscite sono sincronizzate sullo stesso clock, in modo tale che l'intero sistema funzioni come una grande macchina a stati (Fig. 1a). In essa si distinguono due blocchi: uno contenente le logiche booleane ed uno con i registri che "fotografano" lo stato del sistema ad ogni fronte di salita del clock. Lo stato, poi, viene riportato sugli ingressi logici dalla retroazione, mentre l'uscita delle logiche booleane viene stabilizzata prima di essere resa disponibile all'esterno, entro l'intervallo di tempo esistente fra due colpi di clock. Pertanto, non dev'esserci alcuna operazione logica in grado di durare di più (Fig. 1b). In un sistema reale (Fig. 1c), tuttavia, i registri sono distribuiti a bordo di chip che possono contenere milioni di porte logiche. In tal caso, per alcuni segnali di temporizzazione, possono esserci dei percorsi un po' più lunghi rispetto ad altri ed allora occorre ricorrere ad opportuni buffer per far sì che essi trasferiscano alle logiche cui sono destinati la corretta sincronizzazione. Tali buffer, tuttavia, alzano i consumi e non assicurano mai la totale garanzia di non introdurre dei ritardi. Inoltre, la lunghezza delle linee è anche causa di possibili emissioni ed interferenze. Purtroppo tali effetti si possono benissimo manifestare anche in assenza di segnali né sugli ingressi, né sulle uscite.

In un'implementazione asincrona tutte queste linee di clock forzatamente connesse a tutti i registri e a tutte le logiche del sistema non ci sono. Tutte le funzioni logiche sono comandate ugualmente bene, ma in un altro modo. I diversi aspetti del funzionamento asincrono di un cir-

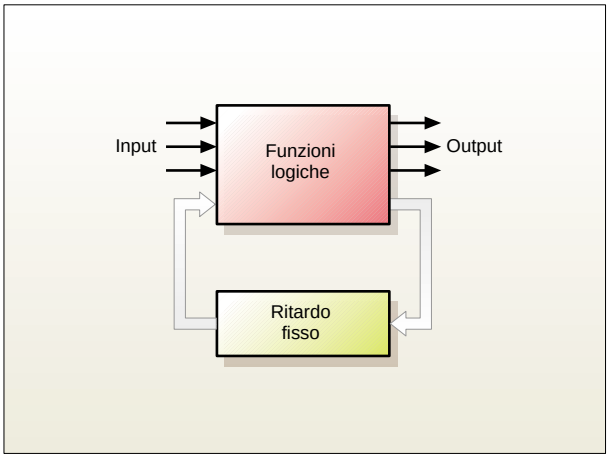

Fig. 2 - Schema asincrono FSM di Huffman

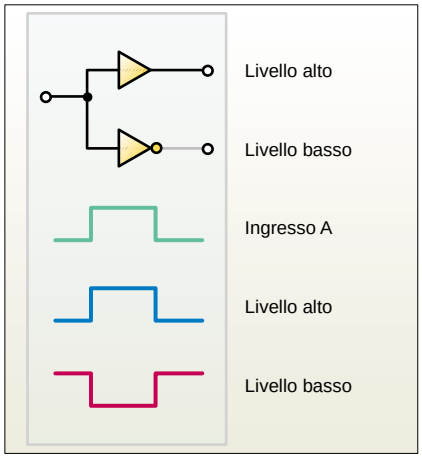

Fig. 3 - Il modello Dual-Rail per il trasporto dei livelli alto e basso del clock su due linee separate

cuito sono riassunti nella tabella 1. Innanzi tutto, si può scegliere se basarsi sui livelli del segnale di clock oppure sui suoi fronti di commutazione. Vi è poi la possibilità di utilizzare due linee diverse per trasportare i livelli alto e basso dei segnali dati, come illustrato nella figura 3: anche se si aumenta l'occupazione di silicio si ha il vantaggio di avere in ogni istante solo una delle due linee attive. Se si hanno nel chip delle logiche che devono funzionare in parallelo, è possibile riunire i rispettivi segnali di clock e farli viaggiare in una linea dedicata. In tutti questi casi si ottiene un sistema indipendente dalle alterazioni della velocità delle porte, principalmente dovute all'irregolarità della tensione di alimentazione e della temperatura. La tecnica maggiormente utilizzata nel disegno delle logiche asincrone si basa sui livelli del segnale di clock e sull'uso delle doppie linee Dual-Rail. In questo caso, è molto semplice utilizzare il ritardo di un segnale nel suo passaggio attraverso la logica combinatoria come clock per i registri di stato grazie al modello di Huffman schematizzato nella figura 2.

L'implementazione

La configurazione Dual-Rail consente di ottenere un sistema con sole linee attive dove, in altre parole, i livelli alti, ovvero i segnali dati attivi, trasportano automaticamente anche il proprio clock, mentre i livelli bassi sono inattivi. Quindi, solo un livello alto, cioè un dato, può causare la commutazione di una porta logica, il suo attraversamento e l'uscita di sé stesso dopo la sua elaborazione logica. Nella figura 4 è illustrata la possibile configurazione Muller C-Element di una porta AND asincrona, dove a sinistra (fig. 4a) si vede la retroazione che serve ad indurre la sequenzialità nelle operazioni combinatorie, mentre a destra (fig. 4b) si vedono i segnali di temporizzazione per la stessa porta. In pratica, quando entrambi gli ingressi sono inattivi l'uscita è bassa ed anche quando appare il primo ingresso alto l'uscita non cambia, mentre cambia il livello della retroazione perciò, al secondo livello alto in ingresso, si potrà vedere l'uscita commutare a livello alto. Se uno degli ingressi diventa inattivo, cioè basso,

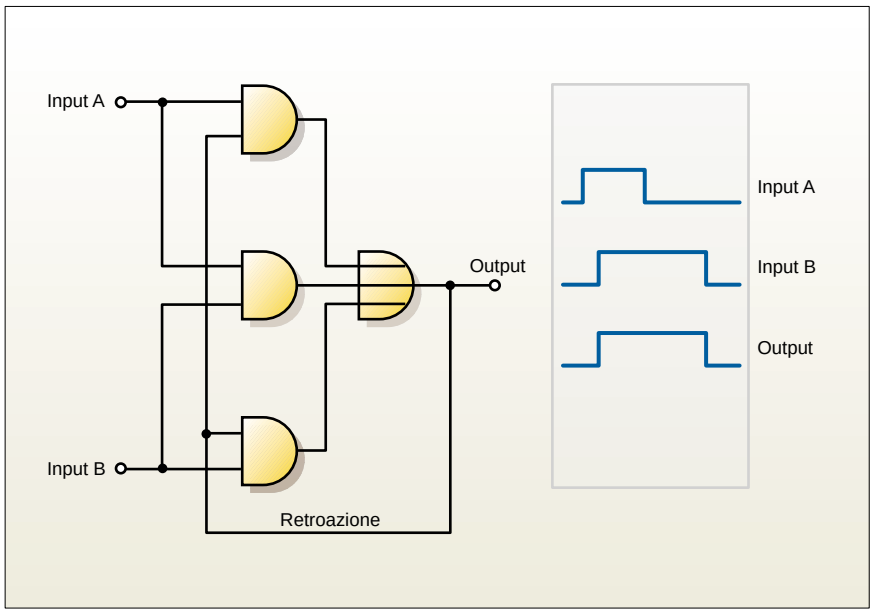

Fig. 4a - Una porta AND asincrona in configurazione Muller C-Element

Fig. 4b - La temporizzazione per una porta AND di Muller

l'uscita resta ferma al livello alto, cioè attiva, grazie alla retroazione, ma se anche il secondo ingresso diventa inattivo, allora l'uscita commuta al livello basso.
D'altra parte, per trasferire i dati fra i blocchi funzionali occorre che i relativi segnali sono stabili. Come si vede nella figura 5, un segnale di Start può essere previsto per indicare un ritardo più lungo del tempo di attraversamento tipico di una logica, in modo tale che un secondo segnale Ready indichi il momento in cui l'uscita della logica è stabile. Da questa configurazione deriva la possibilità di utilizzare il segnale uscente da una logica

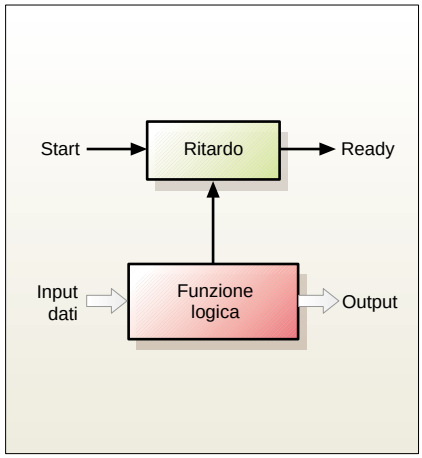

Fig. 5 - Pipeline con linea di ritardo aggiuntiva

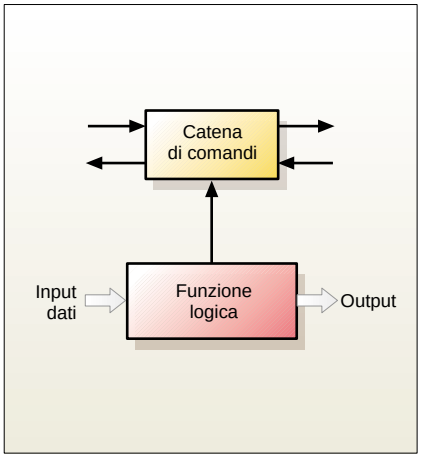

Fig. 6 - Concatenazione in successione dei comandi per il trasferimento dei segnali fra le logiche combinatorie

come segnale di Start per attivare la logica successiva. Così si può fare in modo che quest'ultima mandi il segnale di Ready all'indietro per confermare la fine di un'operazione combinatoria e indurre la continuazione del ciclo, come si vede nella figura 6. Una configurazione leggermente più sofisticata è detta Micro-Pipeline e consiste nel predisporre tre linee di segnali di comando asincroni fra i vari blocchi funzionali del sistema, come si vede nella figura 7, dove per garantire il corretto trasferimento asincrono dei segnali e dei dati sono ugualmente effi-

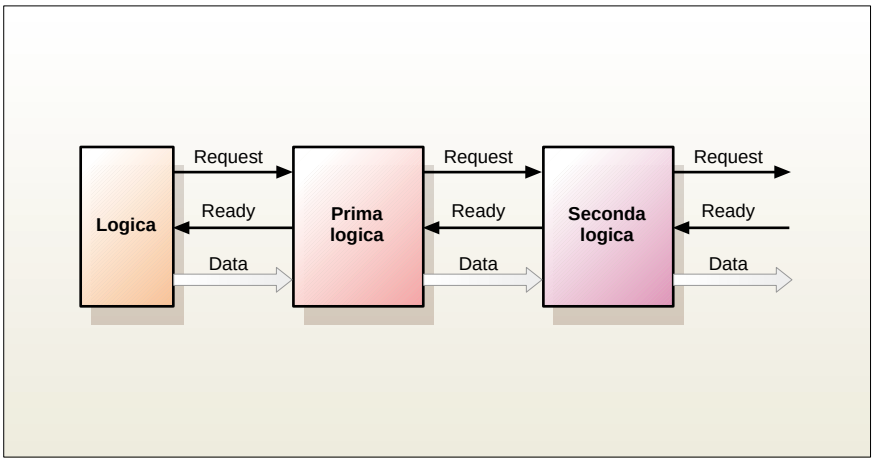

Fig. 7 - La configurazione Micro-Pipeline è la più efficiente nel gestire i trasferimenti asincroni fra le logiche

cienti la tecnica FIFO (first-in first-out) e la LIFO (last-in first-out). La logica che trasmette prepara i dati ed attiva il segnale Request, quindi, la prima logica riceve i dati e conferma con il segnale Ready. Ora, questa effettua l'operazione combinatoria richiestale sui dati e li rende disponibili alla sua uscita. Dopodiché il ciclo si ripete identico dalla prima logica alla seconda e così via di seguito.

In definitiva, l'implementazione asincrona consente di attivare ciascuno dei numerosi blocchi funzionali di un moderno sistema ad alta complessità con una temporizzazione propria ed indipendente da quella degli altri blocchi, in modo tale da poter liberamente scegliere, per esempio, quali blocchi attivare in un dato istante e quali lasciare inattivi ed ottenere così un funzionamento più efficiente del sistema riducendo al tempo stesso i consumi. I vantaggi sono ancor più evidenti quando più blocchi di proprietà intellettuale di fornitori diversi devono coesistere nello stesso sistema: l'implementazione asincrona permette di rispettare i requisiti di ciascuno di essi senza limitarne le prestazioni, anzi, migliorandole. Il beneficio maggiore che offrono le logiche asincrone, dunque, è quello di consentire un evidente contenimento dei consumi nei sistemi con grandi dimensioni, alta velocità e banda larga, nei quali il tradizionale funzionamento sincrono comincia a mostrare qualche limite. È probabile che d'ora in avanti vedremo sempre più System-on-Chip integrare insieme parti sincrone e parti asincrone, con le seconde in progressiva, ma inevitabile crescita rispetto alle prime.

Alcuni link utili per le logiche asincrone:

www.theseus.com
www.synopsys.com
www.exemark.com