

STRUTTURA DI UNA FPGA

*L'analisi
e la conoscenza
della struttura fisica
delle FPGA
è fondamentale
per il loro utilizzo
nello sviluppo
di progetti,
che rispondano
in pieno a quanto
preventivato in fase
di analisi, in termini
di velocità
e prestazioni*

Negli ultimi anni la disponibilità di **TOOLS** e **SW** di sviluppo per circuiti complessi come le **FPGA** ha semplificato enormemente l'attività di progetto su questi prodotti. Si presenta però una situazione anomala, dove l'approccio a questi componenti è quasi diventato più di tipo **SW** che non **HW**. Proprio la conoscenza **HW** di questi componenti ne permette il loro sfruttamento al massimo possibile, sia in termini di performance sia in termini di occupazione di gate. L'articolo vuole evidenziare la struttura fisica di una **FPGA** per comprenderne in linea generale non solo le capacità, ma anche i limiti.

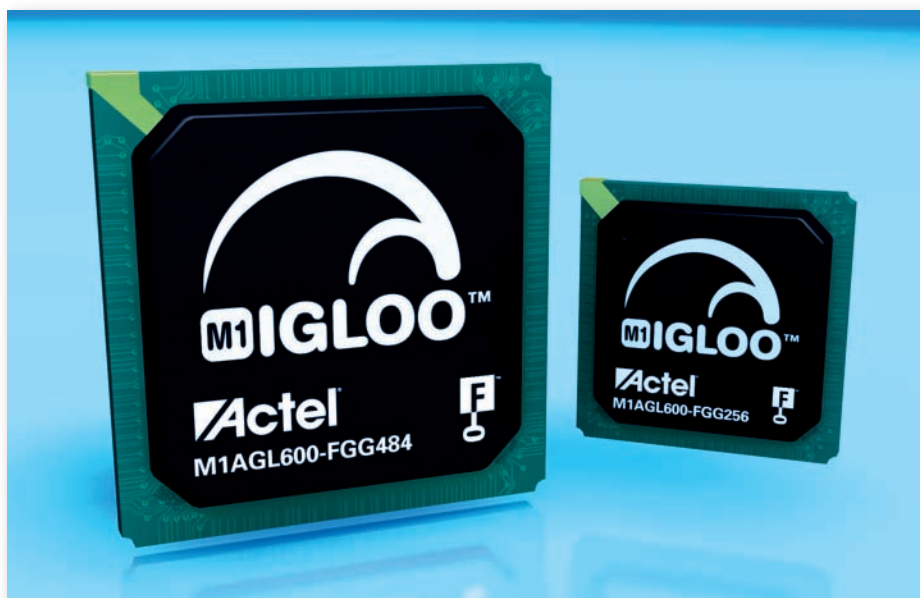
Il componente

La seguente analisi prende come esempio la **FPGA** della famiglia **Proasic3** e **IGLOO** di **ACTEL**. Questa famiglia di prodotti si caratterizza sul mercato per l'uso della tecnologia **FLASH** riprogrammabile, che consente l'operatività immediata al **Power-up**, oltre che consumi di corrente minimi in assenza di **In-rush** iniziale e correnti di quiescenza sotto i 2 mA (**IGLOO** ha correnti di quiescenza di pochi μ A). La disponibilità di densità partendo da

10K di gate per arrivare a 1M di gate e package sia **BGA** sia "normali" **VQFP**, fanno apprezzare questa famiglia che elimina l'antica scelta fra **CPLD** e **FPGA**, almeno in termini di gate da utilizzare. La sua struttura tecnologica è in effetti quanto di più si avvicini oggi all'acronimo originale **Field Programmable Gate Array**, che per tutti è semplicemente **FPGA**.

Struttura di una FPGA

Uno sguardo "inside", oltre i pin aiuta a capire meglio: partendo dal fondo troviamo il **Versatile** che per questa **FPGA** è la cella base (**figura 1**). Attraverso la programmazione dei vari punti di interconnessione è possibile ottenere dalla medesima cella sia un blocco combinatoriale a tre input ed un output, sia un **Flip Flop** di tipo **D** con **Clear** ed **Enable**. Appare quindi evidente in questo primo approccio come una struttura del genere permetta di costruire una qualsiasi rete logica complessa. Ovviamente in questa fase è il tool di sintesi che permette al progettista di costruire, in maniera trasparente, la funzione di trasferimento pensata e per quanto complessa essa sia (ad esempio il core di un microprocessore) il sintetizzatore provvederà alla sua definizione ottimale. La **figura 2** ci consente un approfondimento, osservando come si presenta fisicamente nel componente l'array di celle basi e le sue periferiche integrate. Nella Famiglia **Proasic3** un componente di media densità come **A3P250** o **AGL250** dispone di 6.144 celle basi, pertanto disponiamo di 6.144 **Flip Flop** o porte combinatorie. Appare evidente, sempre dalla **figura 2**, come un compito delicato sia l'interconnessione di queste celle per la costruzione di sistemi o funzioni maggiormente complesse. La fase che sviluppa queste interconnessioni viene definita **Place and Routing**. Per permettere lo sviluppo di tale fase senza perdere in prestazioni, il produttore del silicio ha messo a disposizione del tool di **Place**



and Routing le più ampie risorse possibili, in modo da poter permettere l'interconnessione ottimale. Ogni cella infatti dispone di vari canali di collegamento con le adiacenti, in modo da poter scegliere fra interconnessioni veloci fra celle contigue oppure a gruppi (per esempio nel caso di registri). Insieme a questi collegamenti di tipo locale troviamo le interconnessioni di tipo Global, che sono delle vere e proprie autostrade dove far viaggiare i segnali, come i clock, che dovranno propagarsi per tutto il chip. Per la loro complessità e natura sono sempre in numero limitato e in fase di sviluppo sarà sempre un parametro di cui tener conto come limite di risorsa. Altro blocco che è ormai diventato una presenza costante nelle FPGA è il blocco RAM. Esso è tipicamente configurabile per dimensione e profondità, quindi deve essere visto come un blocco monolitico allocato in una ben definita zona fisica del componente (figura 2). Nel caso della Proasic3 i blocchi di memoria sono allocati nella parte superiore ed inferiore del componente (nella parte inferiore solo per i tagli superiori a 250K gate). Ultimo ma non meno importante è il blocco PLL (in figura 2 segnalato come CCC), che ci permette in questa famiglia di prodotti di generare fino ad un massimo di tre frequenze, partendo da un'unica sorgente di clock e si collega facilmente alle linee di tipo Global, per poter propagare i segnali con il minimo ritardo. Si noti che in figura 2 si rilevano sei locazioni di CCC (Clock Conditioning Circuits) ma solo quella allocata al centro-sinistra è occupata dal PLL, le altre svolgono funzioni solo come linee di ritardo programmabili. A tutto questo si accompagna una suddivisione dei pin di I/O in banchi per permetterne l'utilizzo a diversi tipi di alimentazione, per cui potremo avere l'interfaccia verso un banco SDRAM a 1,8V, mentre l'interfaccia di BUS verso il microprocessore di sistema sarà a 3,3V. Ovviamente l'uso contemporaneo delle due alimentazioni sul medesimo banco non è possibile.

Design e costruzione fisica del progetto

Pur demandando al SW lo sviluppo dei collegamenti in base alle varie funzioni inserite in progetto, appare già evidente che la somma dei tempi di propagazione

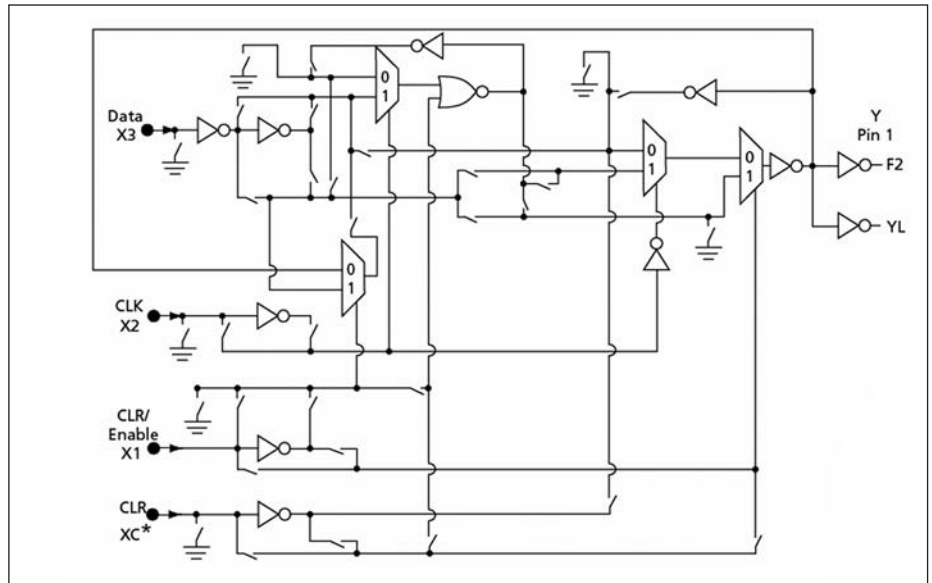


Figura 1: la cella base della FPGA VersaTile.

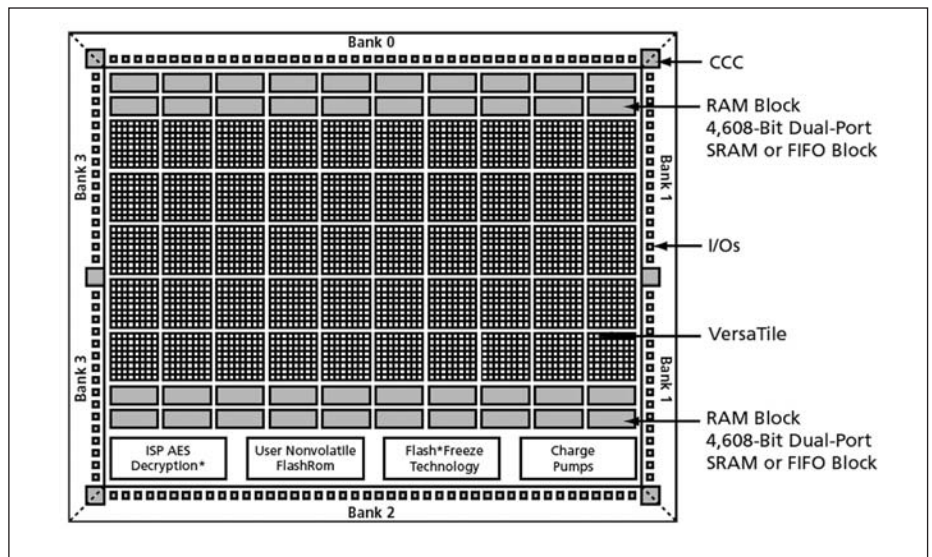


Figura 2: l'array di celle basi e le periferiche integrate.

nelle varie interconnessioni e punti di Routing, oltre che la propagazione dei segnali di tipo globale, pongono un limite alle massime frequenze ammissibili e quindi alle prestazioni ottenibili. Un approccio molto simile a quanto viene normalmente fatto dai progettisti HW, quando si tratta di posizionare i componenti fisici su scheda, può essere un utile metodo. Infatti se proviamo ad immaginare il silicio come una scheda canonica, ci dovremmo sempre porre il problema di dove allocare fisicamente i nostri componenti (in questo caso i moduli VHDL), in modo da ottenere le interconnessioni più corte e il collegamento ottimale fra i vari moduli in

termini di distanza fra loro. Esiste in ogni tool di piazzamento la possibilità, con strumenti più o meno complessi, di definire aree fisiche del componente dove obbligare il SW stesso all'allocazione dei moduli da noi prescelti. Nel caso di ACTEL tale attività viene svolta attraverso una semplice modalità grafica, per cui il progettista, facendo uso del mouse, disegna e definisce le varie aree dove in seconda battuta assegnerà i blocchi specifici, che desidera residenti in una zona ben precisa. In questo modo potremmo allocare fisicamente un intero modulo esattamente adiacente ai blocchi RAM (supponendo per esempio che tale modulo legga e

scriva i medesimi). La **figura 3** riporta il posizionamento obbligato di due moduli: la sezione in rosso è un piccolo microprocessore su FPGA, che è stato allocato adiacente alla zona RAM, mentre la sezione in viola è una periferica del medesimo, che sviluppa quattro canali PWM. Si evidenzia chiaramente come in questo modo le sezioni sono estremamente raccolte e compatte. Un'analisi timing fatta su questo esempio ha evidenziato un miglioramento maggiore del 25% in termini di velocità di esecuzione, rispetto alla prima stesura ove la frequenza massima ammessa era di 97 MHz.

Sfruttamento delle risorse: a volte meglio di no

Definire e allocare parti di progetto in aree di tipo esclusivo, non ha solo un costo in termini di tempo-lavoro per la loro creazione. Infatti creare delle zone esclusive, seppur ottimali dal punto di vista della costruzione fisica, lascia quasi sempre celle inutilizzate. Ma questo ci porta ad una considerazione e ad una domanda estremamente importante per le prestazioni che vogliamo ottenere: quanti gate usare? La risposta sembra ovvia: tutti se si può. In effetti così dovrebbe essere, ma in realtà superata una certa percentuale di occupazione, i tools di Fitting e di Routing, pur lavorando al massimo, non riescono a risolvere in maniera ottimale piazzamenti e interconnessioni. L'astrazione dal progetto porta spesso a dimenticare che le risorse disponibili sul componente sono sempre sufficienti a garantire la completa interconnessione delle celle, ma non le prestazioni. In presenza di questa esigenza, in alternativa ad un passaggio di ottimizzazioni manuali (spostamento di singoli gruppi di celle, creazione di collegamenti preferenziali a maggior velocità, etc.), metodo spesso tedioso e frustrante, è

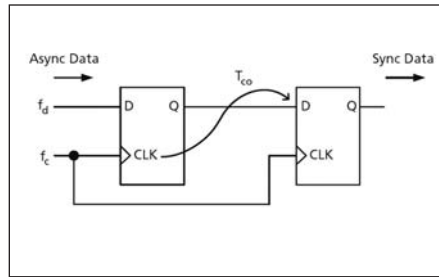


Figura 4: un esempio di aggancio di un segnale esterno da asincrono a sincrono.

consigliabile limitare lo sfruttamento del componente in termini di gate ad una percentuale in grado di lasciare ai tools maggiori scelte e risorse per raggiungere le prestazioni imposte. Una percentuale intorno al 70% rappresenta una riduzione più che ragionevole per qualsiasi tool di Place and Routing. Ovviamente un utilizzo al 100% delle risorse è sempre possibile, a patto di lasciare tempi e prestazioni come valori conseguenti e non primari. Un'applicazione con un clock di sistema di 20-40 MHz nelle famiglie di nuova generazione non avrà mai particolari problemi.

Sincroni o asincroni: una scelta spesso difficile

Questo è un altro dei punti molto controversi durante la stesura dei progetti. Le reti asincrone sono spesso molto più semplici e immediate rispetto alle strutture sincrone, dove si necessita sempre di un clock di sistema. Escluso il tempo di propagazione della rete stessa, la risposta dei blocchi asincroni è immediata. Tuttavia è proprio il clock con i suoi passaggi ben definiti a risolvere i problemi di metastabilità, che affliggono spesso le reti logiche integrate. Infatti, in progetti mediamente complessi è assai facile che un rete sequenziale (per esempio una macchina a stati) possa ricevere un'abilitazione o un segnale di controllo dall'esterno (oppure

da un'altra sezione del progetto) in maniera del tutto asincrona rispetto al clock della macchina stati ricevente e con una cadenza considerevole. Questa situazione porta spesso ad una perdita di controllo della macchina a stati, che di norma si blocca in condizioni non previste. La causa è proprio l'evento asincrono, che in quanto tale, si inserisce senza il rispetto di nessun settling time in una rete che per definizione è di fatto un sistema retroazionato, che decide il suo stato analizzando le variabili d'ingresso, che devono per stabilità rispettare settling ed hold time, come per un qualsiasi Flip Flop. In caso contrario ci troveremo in una condizione non stabile (o metastabile) con risposte della rete non prevedibili. Campionare i segnali di controllo rendendoli sincroni con il medesimo clock della macchina a stati sfasato di 180°, seppur con il consumo di qualche cella, permette il controllo di tutti i tempi di settling and hold, essendo a questo punto noto il tempo utile (mezzo colpo di clock) per il rispetto di tali parametri. La **figura 4** riporta un esempio di aggancio di un segnale esterno da asincrono a sincrono, nell'esempio usando la stessa fase del clock di macchina. Per un interessante approfondimento degli stati metastabili è disponibile nel manuale della famiglia ACTEL Proasic3 un capitolo specifico, che oltre a presentare dettagliatamente l'argomento ne fa una esaustiva caratterizzazione in termini numerici.

Conclusioni

In definitiva considerando le reti sempre più complesse che è possibile inserire in una FPGA (si pensi all'integrazione di interi microprocessori come l'ARM7), non si vuole certo affermare che la costruzione di un progetto complesso in FPGA debba essere sviluppato manualmente. Un approccio però sempre rigorosamente HW è d'obbligo, il poter scrivere la descrizione delle reti in linguaggio C non deve portare ad un approccio troppo "software" dove i tempi di propagazione, settling e hold time sono parametri secondari. I fornitori di silicio mettono a disposizione dei progettisti gratuitamente un ricco gruppo di tools proprio dedicati a queste analisi: usarli è d'obbligo. ■

RIFERIMENTI

■ ■ ■ Proasic3 Data Book http://www.actel.com/documents/PA3_HB.pdf

■ ■ ■ IGL00 Data Book http://www.actel.com/documents/IGL00_HB.pdf

■ ■ ■ Proasic3 App Note : Designing for performance

http://www.actel.com/documents/Design_Performance_AN.pdf

codice MIP