

IMPLEMENTAZIONE DI UN SISTEMA DIGITALE SU DISPOSITIVO XILINX SPARTAN XCS10

RELAZIONE SUL FUNZIONAMENTO E SCHEMI ELETTRICI

Laboratorio di Elettronica C.d.L. Fisica

*Dott. R. Messi
M. Sabene*

CONSIDERAZIONI GENERALI SUGLI FPGA

Un dispositivo **Field Programmable Gate Arrays** consente l'implementazione di circuiti elettronici digitali di tipo combinatorio, sequenziale e di state machines di una complessità che varia dal tipo di dispositivo adottato. In generale un FPGA si può caratterizzare in base al suo numero di CLB ossia di celle configurabili disposte come gli elementi di una matrice i quali permettono, dati dei segnali al loro ingresso di elaborarli in maniera combinatoria e eventualmente di registrarli (due latch D per ogni CLB), altro parametro caratterizzante di un tale dispositivo è il numero di I/O disponibili con i corrispondenti blocchi IOB che permettono di avere il latch di un dato in ingresso o in uscita.

In particolare, un bus di n-bit può essere "latchiato" in ingresso per essere poi utilizzato nel nucleo del dispositivo oppure mandato direttamente in uscita. Nei CLB inoltre è possibile implementare tramite le LUT (vedi figura 2)

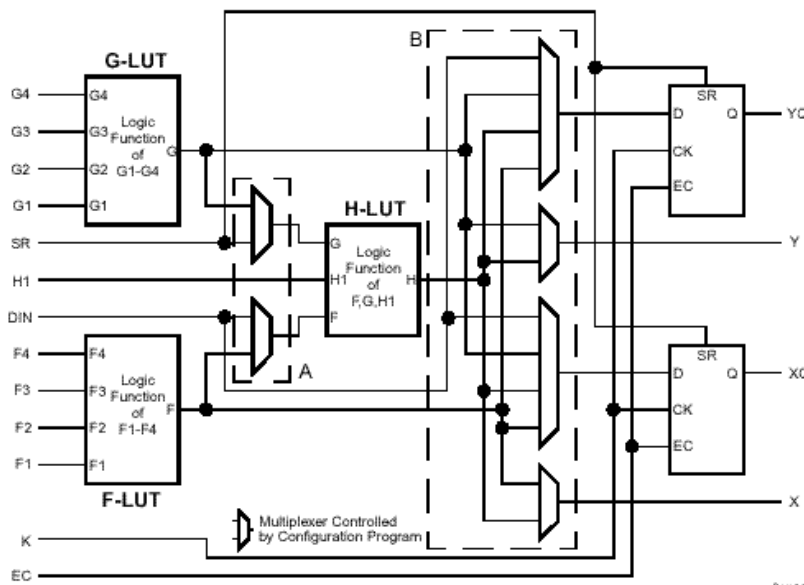


Figure 2: Spartan/XL Simplified CLB Logic Diagram (some features not shown)

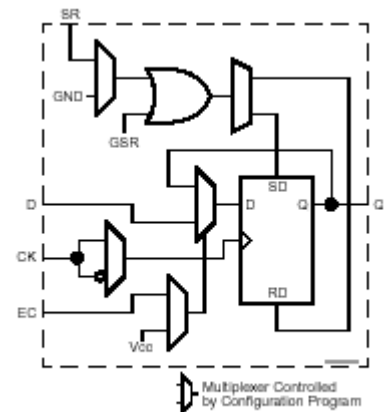


Figure 3: CLB Flip-Flop Functional Block Diagram

delle RAM molto veloci con modalità sincrona di scrittura e asincrona in lettura, senza entrare nel dettaglio, si ha che per ogni CLB si possono configurare delle RAM 16x1, (16x1)x2, 32x2 volendosi indicare con la scrittura (16x1)x2 la possibilità di avere due RAM indipendenti.

I vari CLB e IOB, i dati in ingresso ecc. vengono connessi insieme tramite un sistema di route metalliche e un sistema PSM (Programmable Switching Matrices) che provvede a connettere le varie route per indirizzare il segnale. I PSM usano un dispositivo six-pass Transistor per la connessione tra le varie linee che percorrono in verticale e in orizzontale tutto il dispositivo in particolare le PSM connettono le linee di tipo single-length e double-length (vedi figura 10 a pagina seguente).

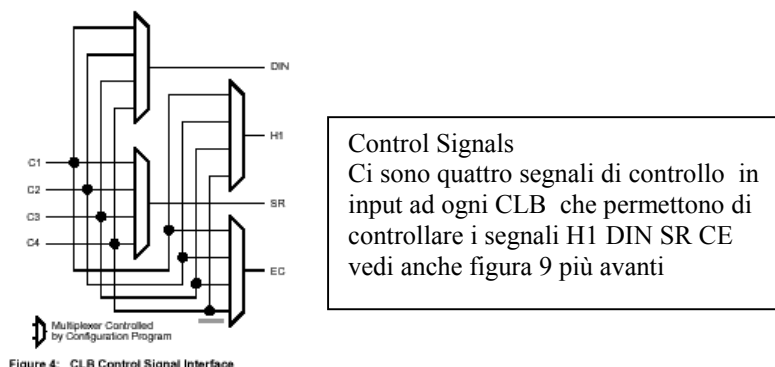


Figure 4: CLB Control Signal Interface

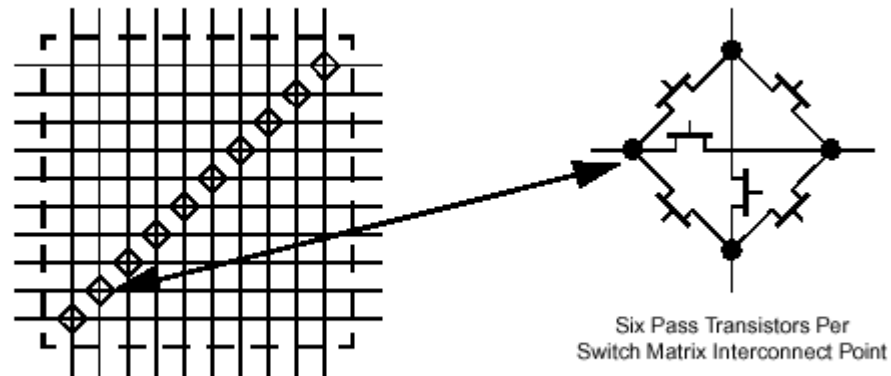


Figure 10: Programmable Switch Matrix

Le route si dividono in:

- CLB routing che corrono lungo ogni riga e ogni colonna della matrice
- IOB routing chiamate VersaRing che formano un anello intorno alla matrice e permettono la connessione tra dispositivi di I/O e le route interne
- Global routing le quali vengono dedicate alla distribuzione dei clocks nel dispositivo con il minimo delay e skew inoltre vengono usate per segnali con un alto fanout

Il software assegna automaticamente il routing basandosi sulle richieste di timing dei vari segnali e in base alla densità del sistema da implementare.

Nel CLB routing si distinguono tre tipi di linee (vedi figura 8) single-length double-length e longlines , come già detto le routing verticali e orizzontali si incrociano nelle PSM che in fase di funzionamento dello Xilinx sono programmate. Il sistema six-pass Transistor si occupa di interconnettere i punti all'interno della PSM .

Nella figura 9 si vede come un CLB è interfacciato con i vari segnali che a loro volta proverranno oppure andranno a finire nel routing

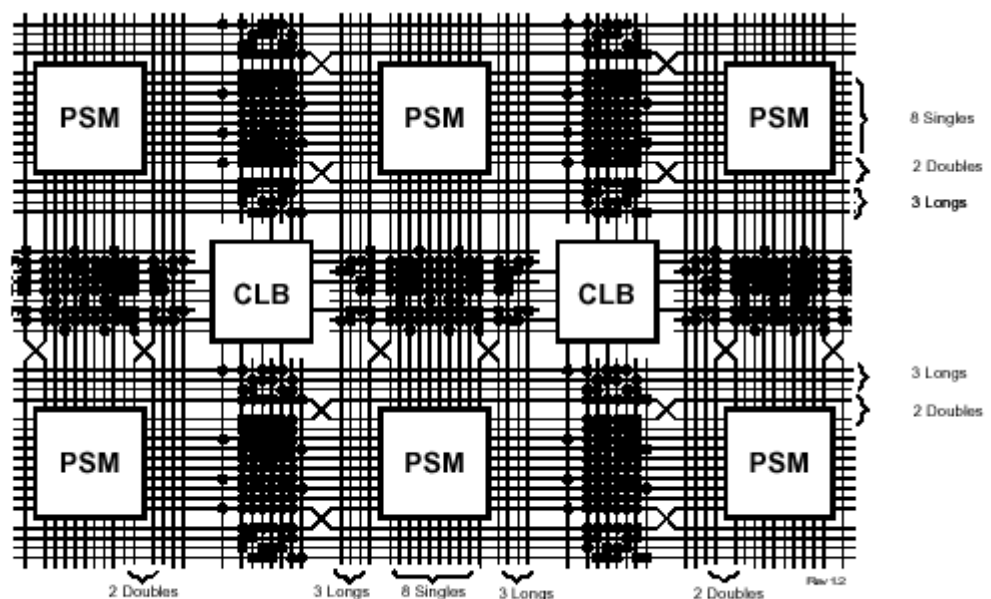


Figure 8: Spartan/XL CLB Routing Channels and Interface Block Diagram

Per avere la massima flessibilità nel routing gli ingressi sono distribuiti intorno al blocco. Il segnale K proviene da un global routing i segnali CIN e COUT che servono a programmare internamente il CLB sono connessi attraverso canali a loro dedicati per non interferire con il routing generale. Come si vede nella figura 8 le single-length lines provvedono a collegare con grande flessibilità CLB adiacenti (8 linee orizzontali) inoltre le single-length verticali connettono le PSM. Poiché questo tipo di linea produce dei delay nel passaggio attraverso le PSM normalmente non viene utilizzata per il routing di segnali su lunghe distanze.

Le double-length lines costituiscono una griglia di segmenti metallici ogni uno dei quali è di lunghezza doppia rispetto alle single-length, queste passano vicino due CLB prima di entrare in una PSM. Inoltre queste linee sono messe in modo (vedi figura 8) da formare delle coppie che possono essere deviate attraverso le PSM, in ogni riga o colonna di CLB. Ci sono 4 orizzontali e 4 verticali double-length lines per ogni CLB e queste permettono un routing molto veloce su brevi distanze.

Le longlines formano una griglia (vedi figura 8) e percorrono in lunghezza e larghezza tutto l'array questo tipo di linea viene usato per segnali con un elevato fan-out o che devono essere distribuiti su lunghe distanze.

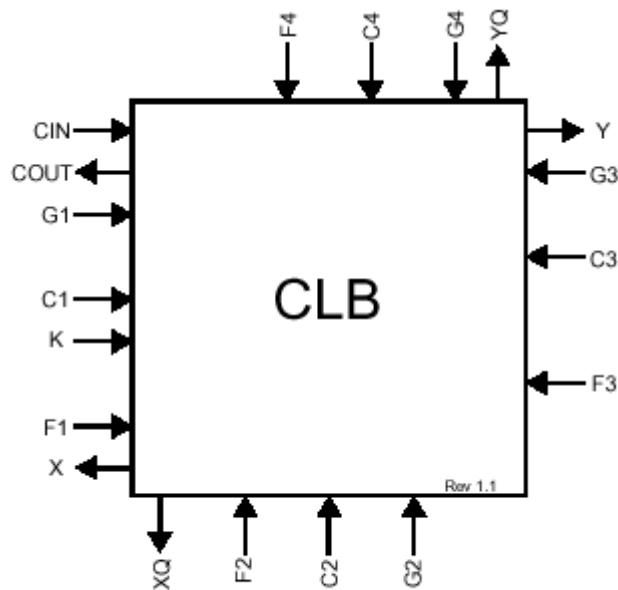


Figure 9: CLB Interconnect Signals

DETTAGLI SULLA SERIE SPARTAN

La figura nella pagina seguente mostra uno schema a blocchi di un FPGA mostrando la posizione relativa di IOB e CLB nei dispositivi Xilinx della serie Spartan si va da un minimo di 100 ad un massimo di 784 CLB e da un minimo di 77 ad un massimo di 224 user I/O.

La Spartan XCS10-4TQ144C usata per questo progetto possiede 196 CLB e 112 IOB con un numero di flip-flops pari a 616 tale dispositivo esiste nelle versioni a 5,5 V e 3.3V: è stato scelto

Table 4: Supported Sources for Spartan/XL Inputs

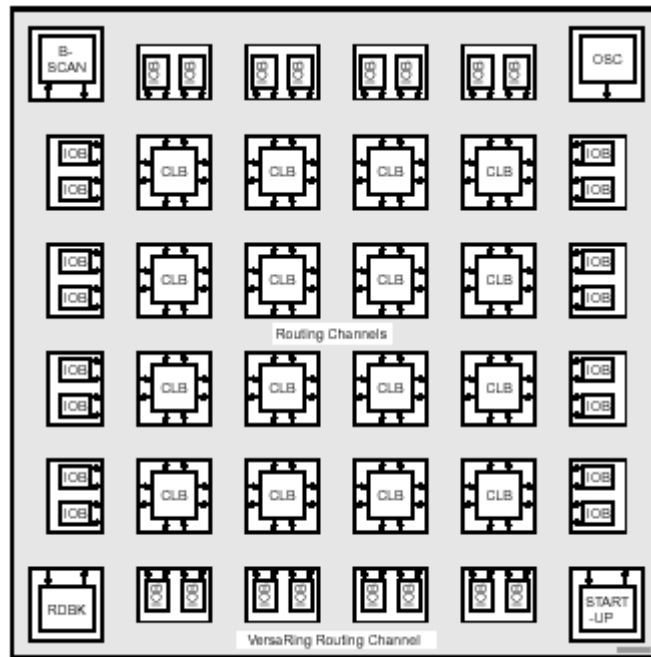
Source	Spartan Inputs		Spartan-XL Inputs
	5V, TTL	5V, CMOS	
Any device, $V_{CC} = 3.3V$, CMOS outputs	✓		✓
Spartan family, $V_{CC} = 5V$, TTL outputs	✓	Unreliable Data	✓
Any device, $V_{CC} = 5V$, TTL outputs ($V_{OH} \leq 3.7V$)	✓		✓
Any device, $V_{CC} = 5V$, CMOS outputs	✓	✓	✓ (default mode)

Table 7: Supported Destinations for Spartan/XL Outputs

Destination	Spartan-XL Outputs		Spartan Outputs
	3.3V, CMOS	5V, TTL	
Any device, $V_{CC} = 3.3V$, CMOS-threshold inputs	✓	✓	Some ¹
Any device, $V_{CC} = 5V$, TTL-threshold inputs	✓	✓	✓
Any device, $V_{CC} = 5V$, CMOS-threshold inputs	Unreliable Data		✓

1. Only if destination device has 5V tolerant inputs

nella versione da 5,5 V dato che questo è lo standard supportato dai dispositivi con cui è interfacciato.



Schema a blocchi di un FPGA

Abbiamo visto precedentemente (vedi figura 2) come è strutturato un CLB della serie Spartan. Ora vorrei mostrare qualche dettaglio un blocco di IOB (vedi figura sotto)

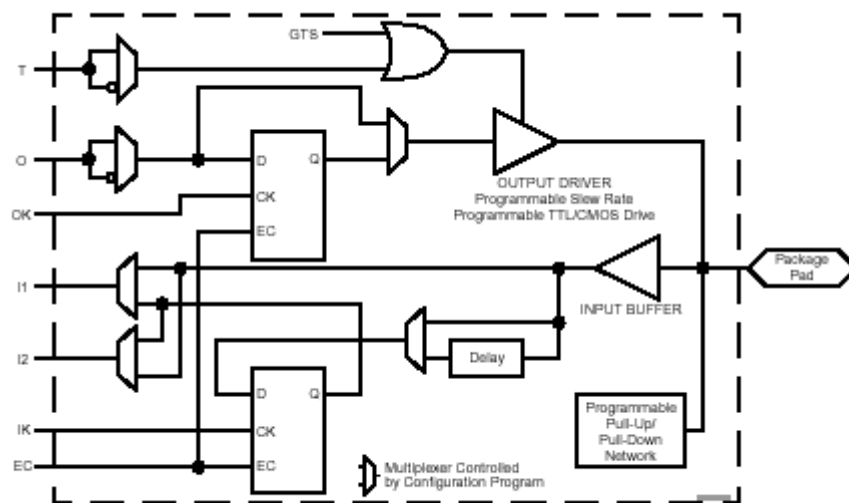


Figure 5: Simplified Spartan/XL IOB Block Diagram

Un segnale in ingresso può essere mandato direttamente nel routing (via I1 e I2) oppure mandato in un input register, in ogni caso il segnale deve passare attraverso un input buffer, il quale in fase di progettazione deve essere inserito nello schematico. In fase di progettazione inoltre si può scegliere la funzionalità del registro scegliendo nelle librerie del dispositivo:

per esempio *piazzando* sullo schematico un IFD si “dice” al Foundation 3.1i di prendere il segnale proveniente da un dato pin di I/O e di indirizzarlo nell’ IFD che svolgerà la funzione di flip flop D, oppure *piazzando* un ILD si otterrà un input latch (transparent-high). E’ possibile inserire un elemento di delay per il segnale in input (vedi figura) , questo elemento viene inserito di default, quindi per avere un segnale senza delay si deve specificare in fase di progettazione. Come si vede dalla figura 5 tramite I1 e I2 il segnale sia esso input diretto o proveniente dal registro viene portato nel routing. L’ input buffer a 5V può essere configurato per un segnale di input a soglia TTL (1.2V)

o CMOS ($V_{cc}/2$), anche i segnali di output possono essere configurati ,indipendentemente da quelli di input. Gli input dei dispositivi Spartan possono essere pilotati da qualsiasi dispositivo a 3.3V configurando l'input in modo TTL (le sources supportate sono mostrate nella tabella 4 a pagina 4). I segnali di output che entrano nell' IOB possono essere mandati direttamente nell'output buffer o possono essere registrati nel flip-flop di output per poi essere mandati nell'output buffer. Le funzionalità di questi flip-flop come le funzionalità dei registri di input sono mostrate nelle tabelle 3 e 6.

Table 3: Input Register Functionality

Mode	CK	EC	D	Q
Power-Up or GSR	X	X	X	SR
Flip-Flop		1*	D	D
	0	X	X	Q
Latch	1	1*	X	Q
	0	1*	D	D
Both	X	0	X	Q

Legend:
X Don't care
 Rising edge (clock not inverted)
SR Set or Reset value. Reset is default.
0* Input is Low or unconnected (default value)
1* Input is High or unconnected (default value)

Table 6: Output Flip-Flop Functionality

Mode	Clock	Clock Enable	T	D	Q
Power-Up or GSR	X	X	0*	X	SR
Flip-Flop	X	0	0*	X	Q
		1*	0*	D	D
	X	X	1	X	Z
	0	X	0*	X	Q

Legend:
X Don't care
 Rising edge (clock not inverted)
SR Set or Reset value. Reset is default.
0* Input is Low or unconnected (default value)
1* Input is High or unconnected (default value)
Z 3-state

DESCRIZIONE DEL SISTEMA IMPLEMENTATO

Si tratta di un sistema digitale che genera segnali di controllo e clock per un altro sistema chiamato A-board (che vedremo più in dettaglio dopo) provvede inoltre a serializzare un dato a 16 bit da inviare sempre alla A-board. come si vede nello schema a blocchi lo Xilinx risiede nella B-board insieme al generatore di master CLOCK.

In ingresso abbiamo innanzitutto un DATAIN a 22 bit in cui vengono immessi tutti i tipi di dati necessari che verranno registrati nel registro dati (vedi schema) a questo punto il dato è pronto a essere utilizzato nel blocco opportuno oppure mandato in OUT. Per esempio se si vuole caricare una locazione di memoria nella A-board, dopo averla indirizzata, si scrive il dato nel DATAREG, lo si carica nella locazione indirizzata tramite il comando LPULMEM.

Vediamo altri esempi di funzioni.

Il BUS a 4 bit del CODE attiva le linee del decodificatore, per esempio CODE = 0000 attiva le linee di PRESET che azzerano tutti i registri presenti nel sistema oppure CODE = 0001 attiva la linea di LREGDATA che abilita alla lettura il registro dati. Il decodificatore è sottoposto ad uno STROBE attivo alto. Solo quando lo STROBE viene attivato, viene attivata la linea corrispondente al codice in ingresso.

Come segnali di ingresso abbiamo inoltre un RESET che può essere attivato direttamente dall'esterno, (con le stesse priorità del PRESET attivato da codice) e uno STOP proveniente direttamente da PC ma che, come si vedrà in seguito è generato dalla A- board alla fine di ogni "evento".

In uscita abbiamo un bus 22 bit che trasporta il dato da caricare nelle varie locazioni di memoria della A- board e un bus a 11 bit che trasporta l'indirizzo di memoria.

Dato che le A-board sono 22 , il CDEC indirizza anche la A- board della quale caricare le memorie quindi si ha per esempio :

indirizzamento A- board , indirizzamento locazione di memoria , caricamento della locazione; ovviamente il dato che si carica nella locazione di memoria sarà costituito dai primi 11 bit del DATAIN (vedi schema) .

Un altro esempio di segnale in uscita è il DECDDATA un segnale seriale costituito da una sequenza di 16 bit che va a caricare un DAC nella A- board, il DACDATA insieme al DACCLK che è un clock sottomultiplo in frequenza del MCLK, e all'ENDLD vengono generati nel blocco SRL1 (vedi schema SRL1).

Il DACCLK ci dà la *velocità* con cui il dato viene serializzato e inviato nel DAC.

Altri segnali di uscita sono generati nel blocco CLKCONT (vedi schema) il più importante dei quali è l'O_CLK che rappresenta il master CLK diviso, questo viene mandato nella A- board come CLK principale dei vari eventi.

Per esempio una tipica fase di azione consiste nel caricare tutte le memorie della A- board, caricare tutti i DAC, e quindi caricare il valore per cui si vuole dividere il MCLOCK e fare partire l'evento; a livello dello Xilinx far partire un evento significa mandare in uscita l'O_CLK e abilitare altri segnali di controllo come N_CNT_EN e LENMOD.

Per concludere la rassegna dei segnali in uscita abbiamo O_RST che viene generato ad ogni segnale di STOP e *reseta* il contatore della A_ board.

DETTAGLI SUI VARI BLOCCHI

DEC1

E' scritto in VHDL (vedi listato) e provvede a decodificare il CODE a 4 bit in ingresso, ha 15 linee di uscita che vengono attivate per un tempo pari a quello in cui il segnale STROBE è alto, quando STROBE = 0 le uscite sono tutte a 0. Le uscite del DEC vanno ad abilitare i vari blocchi : DATAREG, REGADD, SRL1, REPERT, CLKCONT, e all'interno dei blocchi posso attivare diverse funzioni.

RDATA

(vedi schema) ed è un LATCH a 22 bit che viene abilitato dalla linea LREGDATA del DEC.

Questo LATCH è stato implementato utilizzando degli ILD che sono dei D-flip-flop che si trovano nei dispositivi IOB in questo modo si cerca di sfruttare il più possibile le risorse presenti nelle celle di periferia IOB.

RADD

Questo blocco è un LATCH a 16 bit utilizzato per indirizzare le varie locazioni di memoria presenti nelle A- board e per indirizzare i vari DAC, per implementare questo tipo di registro (vedi schema) sono stati utilizzati degli OFD che sono dei D-flip-flop presenti negli IOB che in questo caso si è cercato di ottimizzare lo sfruttamento delle risorse dello Xilinx.

CLOCKGEN

Il CLOCKGEN si può ritenere la parte centrale del sistema, in questo blocco viene generato l'O_CLK. Questo clock rappresenta un sottomultiplo in frequenza del MASTERCLOCK infatti all'interno del CLOCKGEN viene effettuata la divisione mediante il blocco DIV (vedi schema) del MASTERCLOCK. Questa divisione viene effettuata caricando il registro a 20 bit del DIV con il divisore, ottenendo in uscita dal DIV un segnale con DUTYCYCLE =1. Il DIV permette di effettuare una divisione per $D = 1$; $D = 2$; $D = 2^{(n+1)}$. Il divisore viene caricato esternamente tramite il DATAIN a 22 bit di cui 20 costituiscono 2^n e 2 bit servono per la divisione $D = 1$ e $D = 2$. Questo sistema di divisione come del resto tutti i sistemi che sono stati implementati rispondono ad una logica che si adatta alle esigenze di uno sviluppo di un software in C che dovrà sovrintendere al funzionamento globale dello Xilinx e di tutti i dispositivi con cui è interfacciato (A- board scheda di acquisizione National Instruments). Il CLOCKGEN inoltre si occupa anche di generare una sequenza di clock detta N-STEP: si carica il numero di cicli di clock nel COMP_BLOCK il quale manda in uscita un numero predeterminato di cicli con una frequenza opportunamente divisa nel DIV. Sempre nel CLOCKGEN abbiamo il DAC_CLK_DIV (vedi schema) che genera un segnale di clock anche esso sottomultiplo in frequenza del MASTERCLOCK in particolare si ha una divisione per $D = 1$; $D = 2$; $D = 4$; $D = 8$ con un DUTYCYCLE =1. Il segnale DAC_CLK generato viene portato in ingresso a SRL1 il serializzatore di dati.

SRL1

In questo blocco viene serializzato un dato a 16 bit costituito appunto dai primi 16 bit del DATAIN. In ingresso all'SRL1 (vedi schema) abbiamo , oltre al dato da serializzare e LDACMEM per l'abilitazione, il clock che come abbiamo visto prima è il DAC_CLOCK. Questo clock permette al dato in ingresso al blocco SERIAL (vedi listato in VHDL) di essere serializzato con una velocità uguale alla sua frequenza .

Il dato serializzato viene posto in uscita dallo Xilinx per essere caricato nel DAC (precedentemente indirizzato) nella A- board.

In questo blocco viene generato un segnale di controllo ENDLD che dura tutto il tempo della serializzazione, questo serve per far sì che si abbia un istante di inizio e un istante di fine per la serializzazione stessa.

REPEAT

In questo blocco (vedi schema) viene implementato un sistema che permette la ripetizione di una data *sequenza* m volte, a livello dello Xilinx questo vuol dire che l'O_CLK che in generale guida gli eventi nelle A- board, viene ripetuto ogni volta che arriva un segnale di STOP, vedremo meglio questo argomento nella descrizione delle funzioni vere e proprie dello Xilinx nei confronti dei dispositivi ad esso interfacciati, inoltre nel REPEAT si può caricare una pausa tra una sequenza di clock e l'altra, questo avviene nel COMP_BLOCK1. In uscita dal REPEAT abbiamo il segnale STOPREP che viene portato in uscita dallo Xilinx e si occupa di *resettare* il contatore delle memorie nella A- board e di fermare definitivamente il processo di ripetizione.

INTERFACCIAMENTO E FUNZIONE DELLO XILINX

Lo Xilinx alloggiato sulla B-board (vedi schema blocchi) è interfacciato da un lato con una scheda di acquisizione da PC National da cui preleva i segnali (TTL a 5 V) come il DATAIN, il CODE lo SROBE, il RESET e lo STOP (MASTER-CLOCK viene generato su un dispositivo clock generator allocato sempre sulla B-board), dall'altro le sue uscite sono interfacciate con 22 A-board (vedi schema a blocchi) in parallelo anche in questo caso si tratta di dispositivi TTL. Una A-board (vedi schema a blocchi) è un dispositivo costituito da 20 DAC, da una memoria con 2 alla 11 indirizzi, i quali mandati in sequenza nei DAC costituiscono quello che viene definito evento. I dati delle varie locazioni di memoria vengono presentate ai DAC in sequenza, dove la sequenza è data dal COUNTER il cui clock è esattamente l'O_CLK che esce dallo Xilinx. Se ad esempio sulla MEMXX vengono caricate $n < 2^{11}$ locazioni un evento consisterà nel presentare all'ingresso di ogni DAC una sequenza di n bit con una velocità che è quella della O_CLK. Le MEMXX vengono caricate in modo che l'ultima locazione di memoria contenga una stringa di 1 di modo che l'arrivo di questa stringa segnali la fine dell'evento infatti ogni dato prelevato dalle MEMXX viene prima mandato in una AND a 20 bit la presenza di una stringa di 1 genererà un segnale che per lo Xilinx sarà il segnale di STOP (in particolare questo segnale viene mandato al PC e poi da questo rimandato nello Xilinx) . il segnale di STOP fa sì che lo Xilinx termini la generazione O_CLK. Un tipico esempio di evento, detto evento FREE RUNNING consiste :

1. Caricamento delle memorie XX di ogni A – board per fare questo viene tramite RADD indirizzata una A – board , una volta indirizzata , vengono indirizzate una ad una tramite PULBUS a 11 bit le varie locazioni di memorie , le quali vengono caricate con il bus DATAOUT dello Xilinx.
2. Caricamento dei DAC, i DAC vengono indirizzati uno alla volta tramite CDECBUS per l'indirizzamento della colonna e tramite i primi 5 bit di PULBUS per l'indirizzamento della riga. Una volta indirizzato un DAC questo viene caricato tramite un serializzatore con un dato che gli indichi la corrente da far passare nel suo LED, in pratica gli dice l'intensità luminosa con la quale il LED deve accendersi.

3. Caricamento del divisore di clock si pone in ingresso al DATAIN il valore in binario del divisore del MASTER-CLOCK a questo punto tutto è pronto per dare l'avvio all'evento.
4. Tramite CODE=6h si dà il via all'evento l'O_CLK comincerà la sua corsa facendo correre anche il COUNTER della A-board (il quale viene ad essere abilitato tramite N_CNT_EN, il COUNTER indirizzerà in sequenza tutte le locazioni di memoria della sua A-board generando l'evento).
5. Quando il COUNTER indirizzerà l'unica stringa con tutti 1 questa segnerà la fine dell'evento generando lo STOP che fermerà l'O_CLK e manderà lo Xilinx in STANDBY.

È possibile durante il RUN dell'evento effettuare una pausa: tramite CODE=Bh si attiva il WAIT ENABLE, questo è un toggle e mediante CODE=6h è possibile in qualsiasi momento far ripartire l'evento. Nei grafici delle simulazioni si può vedere in dettaglio la sincronizzazione di tutti i segnali, in particolare si può notare che ogni azione dettata dal CODE è subordinata ad uno STROBE che ha la durata di molti cicli di clock. Un altro esempio interessante di funzione dello Xilinx è quello di permettere di ripetere n volte un evento come quello descritto sopra con un intervallo di tempo tra un evento e l'altro che può essere definito, in particolare l'intervallo tra due ripetizioni può andare da un minimo di 0,1s ad un massimo di 5s questo range è dettato dalle necessità dell'ambito in cui il sistema si inserisce. Un'altra funzione ancora consiste nel poter far eseguire all'evento N-STEP (con N caricabile) alla volta: se abbiamo un evento costituito da 100 STEP (perché abbiamo caricato solo 100 locazioni di memoria) possiamo suddividerlo ad esempio in 10 eventi da 10 STEP, il sistema, compiuti i primi 10 STEP, attenderà un segnale dall'esterno (CODE=Ah) per ripartire ed eseguire i successivi 10 STEP e così via fino alla fine dell'evento che come sempre sarà indicato con una stringa di 1. Altra funzione fondamentale è costituita dalla funzione di SINGLESTEP che permette di fare il DEBUG dell'evento: quando viene attivato SINGLESTEP, l'O_CLK che esce dallo Xilinx segue il profilo del segnale di STROBE che viene inviato dal PC quindi si può dare un periodo al clock arbitrario.

CENNI SULLA SIMULAZIONE E IMPLEMENTAZIONE

Questo sistema, è stato simulato e implementato mediante il tool di sviluppo Foundation 3.1i, il quale è un software sviluppato dalla Xilinx e dedicato ai suoi dispositivi. Alcuni circuiti sono stati scritti in VHDL che è un linguaggio molto potente per la descrizione hardware ad alto livello, inoltre è un linguaggio totalmente indipendente dal dispositivo usato: un circuito scritto in VHDL viene compilato e viene creato un blocco logico che si può inserire in una struttura gerarchica superiore, come si vede negli schemi la descrizione dell'hardware avviene in maniera gerarchica.

Parallelamente alla descrizione, l'hardware può essere simulato in maniera funzionale, quindi considerando i vari elementi (gate, flip-flop) come ideali. (vedi simulazioni)

La fase di implementazione infine traduce fisicamente, tramite un flow-engine, sul dispositivo scelto, il circuito creando un file, *file.bit*, durante la fase di progettazione si deve tenere bene presenti quali sono i limiti fisici di spazio e il numero di pin necessari, e si deve cercare di sfruttare al massimo anche la periferie dell'array altrimenti si rischia di usare solo il nucleo e potrebbe non essere sufficiente il numero dei flip-flop. È possibile lanciare il motore di implementazione avendo impostato dei *constraints* ossia è possibile impostare dei valori per esempio sui tempi di ritardo di un dato segnale, in questo caso il programma terrà conto di questo facendo percorrere al segnale una route più lunga.

Vorrei concludere accennando brevemente alla fase di programmazione che rende funzionante il dispositivo, una volta implementato il circuito, il tool di sviluppo crea una net list la quale "piazza" e collega i vari componenti, inoltre da questa viene generato un *file.bit*: ogni bit definisce lo stato di una LUT oppure di un multiplexer oppure di un six-pass Transistor. Questo file ha un formato adatto ad essere caricato sulla PROM (di tipo seriale: per XCS10 serve la PROM 17S10 con 95008 bit). Al momento del power-up viene automaticamente avviata la Configuration Sequence

mediante la quale si ha la comunicazione (IEEE 1149.1) tra Xilinx e PROM che carica serialmente il programma: dopo lo Xilinx è operativo.

SPECIFICHE SPARTAN

Spartan Absolute Maximum Ratings*

Symbol	Description	Value	Units
V_{CC}	Supply voltage relative to GND	-0.5 to +7.0	V
V_{IN}	Input voltage relative to GND (Note 2, 3)	-0.5 to $V_{CC} + 0.5$	V
V_{TS}	Voltage applied to 3-state output (Note 2, 3)	-0.5 to $V_{CC} + 0.5$	V
T_{STG}	Storage temperature (ambient)	-85 to +150	°C
T_{SOL}	Maximum soldering temperature (10s @ 1/16 in. = 1.5 mm)	+280	°C
T_J	Junction temperature	Plastic packages +125	°C

Spartan Recommended Operating Conditions

Symbol	Description		Min	Max	Units
V _{CC}	Supply voltage relative to GND, T _J = 0°C to +85°C	Commercial	4.75	5.25	V
	Supply voltage relative to GND, T _J = -40°C to +100°C	Industrial	4.5	5.5	V
V _{IH}	High-level input voltage	TTL inputs	2.0	V _{CC}	V
		CMOS inputs	70%	100%	V _{CC}
V _{IL}	Low-level input voltage	TTL inputs	0	0.8	V
		CMOS inputs	0	20%	V _{CC}
T _{IN}	Input signal transition time			250	ns

Note 1: At junction temperatures above those listed as Recommended Operating Conditions, all delay parameters increase by 0.35% per °C.

Note 2: Input and output measurement thresholds are: 1.5V for TTL and 2.5V for CMOS.

Spartan DC Characteristics Over Operating Conditions

Symbol	Description	Min	Max	Units
V_{OH}	High-level output voltage @ $I_{OH} = -4.0\text{ mA}$, V_{CC} min	TTL outputs	2.4	V
	High-level output voltage @ $I_{OH} = -1.0\text{ mA}$, V_{CC} min	CMOS outputs	$V_{CC} - 0.5$	V
V_{OL}	Low-level output voltage @ $I_{OL} = 12.0\text{ mA}$, V_{CC} min (Note 1)	TTL outputs	0.4	V
		CMOS outputs	0.4	V
I_{CCO}	Quiescent FPGA supply current (Note 2)	Commercial	3.0	mA
		Industrial	6.0	mA
I_L	Input or output leakage current	-10	+10	µA
C_{IN}	Input capacitance (sample tested)		10	pF
I_{RPD}	Pad pull-up (when selected) @ $V_{IN} = 0\text{V}$ (sample tested)	0.02	0.25	mA
I_{RPD}	Pad pull-down (when selected) @ $V_{IN} = 5\text{V}$ (sample tested)	0.02		mA

Note 1: With 50% of the outputs simultaneously sinking 12 mA, up to a maximum of 64 pins.

Note 2: With no output current loads, no active input pull-up resistors, all package pins at V_{CC} or GND, and the FPGA configured with a Tie option.

