Implementazione in VHDL di un

Modulo per la gestione dei convertitori audio

Su schede: Xess XSA-50
Xess XTD-2

Christian Gregorutti

1 Introduzione

Questo progetto si occupa di pilotare i convertitori audio presenti sulla scheda Xess XSA-50, convertire tramite opportuni registri il segnale seriale d'ingresso in pacchetti da 16 bit (pronti per una successiva elaborazione), quindi rigenerare un segnale seriale e portarlo in uscita.

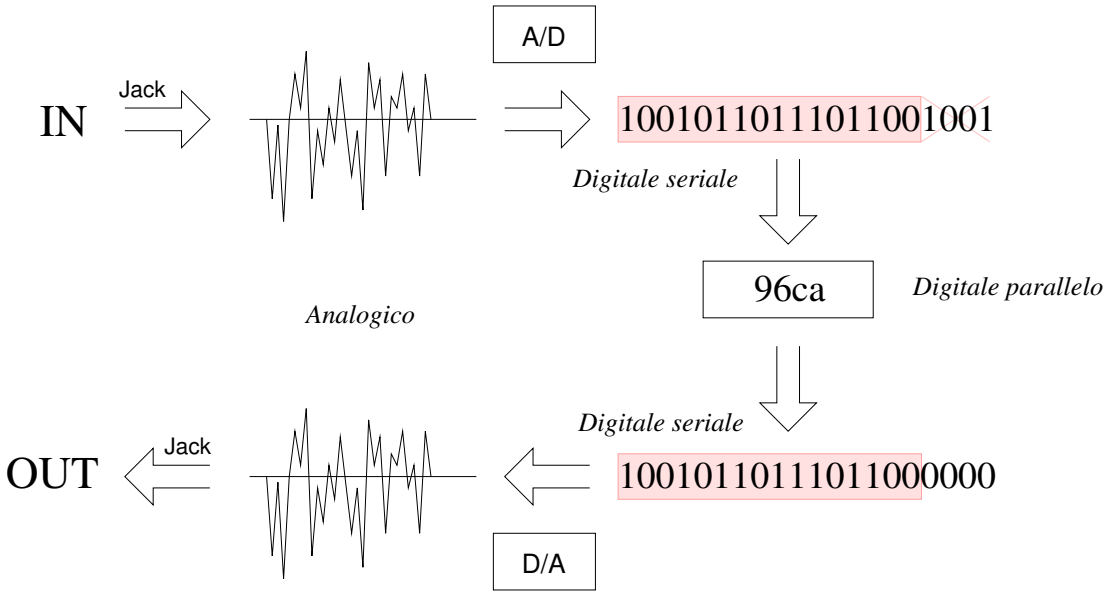


Figura 1: Schema logico del progetto

2 Il convertitore AK4520A

La scheda in dotazione possiede un convertitore stereo AK4520A che accetta in ingresso due segnali analogici (canale destro e sinistro) dal jack J1, converte il segnale da analogico a digitale e spedisce il segnale digitalizzato all'FPGA in modo seriale; in maniera duale accetta dall'FPGA un flusso seriale di bit e lo converte nei due segnali digitali in uscita che escono dalla scheda attraverso il jack J2 (dopo essere stati amplificati da un amplificatore integrato).

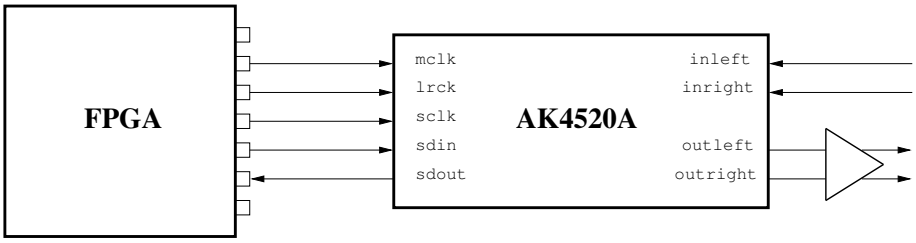


Figura 2: L'AK4520A

2.1 I clock interni

Il convertitore in questione utilizza tre clock: `lrck`, `sclk` e `mclk` pilotati dall'FPGA:

- `lrck` è il segnale che seleziona il canale sinistro (0) o destro (1): questo clock determina la frequenza di campionamento in quanto dice al convertitore quando prendere ogni campione;
- `sclk` è usato per sincronizzare le sequenze di bit: deve avere almeno tanti cicli (per canale) quanti il bitrate di ogni campione prelevato, il che significa che se campiono a 20 bit dev'essere almeno 40 volte la frequenza di campionamento;
- `mclk` è il master clock che viene usato per sincronizzare le operazioni interne al convertitore.

NOTA: Benché l'AK4520A supporti quattro modi diversi di gestire i dati seriali attraverso i suoi pin (`DIF1` e `DIF0` per il bitrate, `CMODE` per la frequenza di campionamento e di conseguenza per il master clock), la scheda non dá la possibilitá di agire su questi piedini avendoli giá collegati il primo a DVDD, il secondo e il terzo a massa (si vedano i data sheet). Di conseguenza i vincoli sono:

- `sdto` (ADC): 20 bit, MSB justified;
- `sdti` (DAC): 20 bit, MSB justified;
- `sclk` $\geq$ 40fs (fs = frequenza di campionamento);
- `mclk`: 256fs.

In particolare in questo progetto si userá un `sclk` pari a 64fs, quindi i vari clock avranno un valore di:

- `clk`: 50MHz (il clock globale del progetto);
- `mclk`: 12.5MHz (256fs);
- `sclk`: 3.125 MHz (64fs);
- `lrck`: 48.828KHz (fs).

2.2 La codifica del segnale

Come si è già detto, il segnale esce dal convertitore A/D trasformato in un treno di 20 bit, ordinati dal più significativo al meno significativo. La figura 3 mostra la sincronizzazione di questo processo:

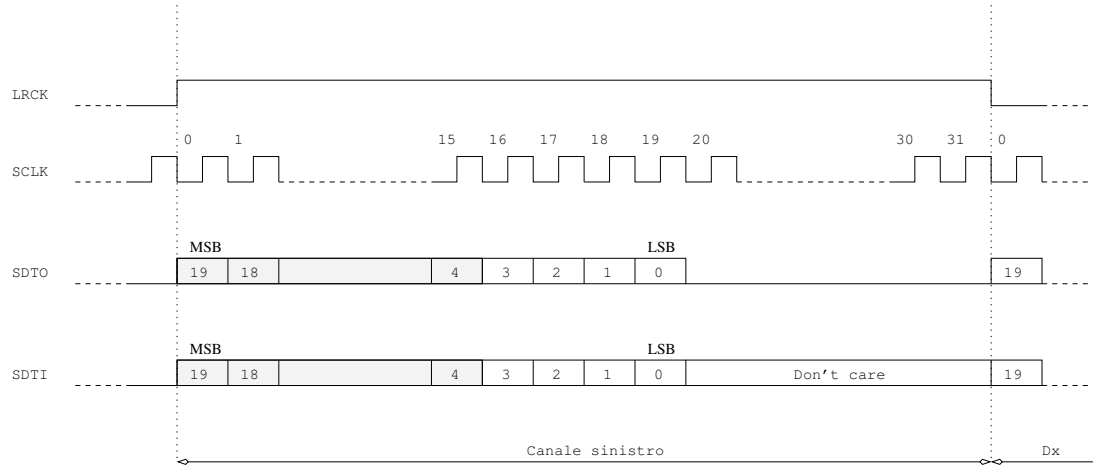


Figura 3: La sincronizzazione dei convertitori

- il segnale lrck passa allo stato logico alto, quindi i convertitori di ingresso e di uscita si occupano del segnale sul canale sinistro;

- il treno di 20 bit inizia ad uscire dal convertitore A/D non appena lrck passa da uno stato logico es. basso (canale destro) a quello alto (canale sinistro);

- allo stesso modo il convertitore D/A in uscita si aspetta che il treno di bit inizi ad arrivare nel momento in cui lrck cambia stato;

- i bit entrano (ed escono) sincronizzati con il segnale sclk.

Ogni fs^{-1} secondi il convertitore A/D campiona la linea d'ingresso e codifica il campione su 20 bit in complemento a 2. Il convertitore codifica con 7FFFFH(@20bit) ogni tensione pari o maggiore a 5V e con 80000H(@20bit) ogni tensione pari o minore a -5V; idealmente l'assenza di segnale dovrebbe venir codificata con 00000H(@20bit). L'offset in continua viene rimosso dal filtro passa-alto interno al AK4520A.

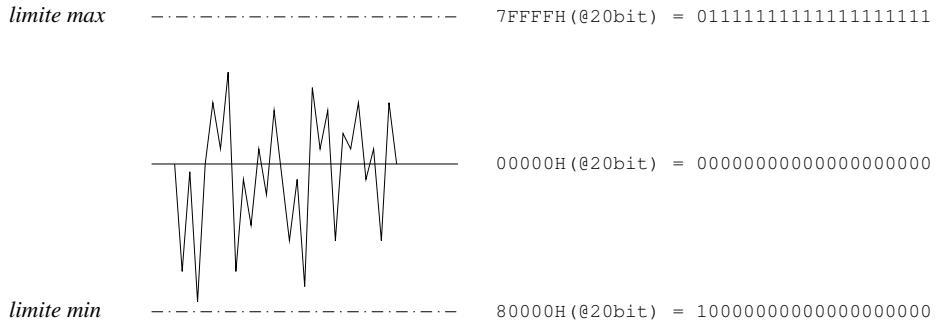


Figura 4: La codifica in complemento a 2

2.2.1 Frequenza di campionamento e bitrate

Ci sono due caratteristiche del progetto che possono sembrare inusuali: la frequenza di campionamento a 48.828 kHz (anziché i classici 44.100) e la decisione di utilizzare solo 16 dei 20 bit messi a disposizione dai convertitori. Queste due caratteristiche sono frutto di una scelta esclusivamente pratica:

- generare un clock di 48.282 kHz a partire da un master clock di 50 MHz è estremamente più semplice che generarne uno di 44.100 kHz: nel primo caso infatti è sufficiente dividere il clock per una potenza di due ($50000000/1024 = 48828.125$); per i dettagli di implementazione si rimanda al paragrafo 3.1;

- la scelta di utilizzare solo 16 bit deriva dal fatto (banale) che risulta molto più comodo lavorare con 2 byte anziché con due byte e mezzo... (si pensi ad esempio alle memorie SDRAM presenti sulle scheda XSA, organizzate in celle da 16 bit); in questo caso vengono trascurati i 4 bit meno significativi, forzati a zero all'uscita (come mostrato in figura 1).

3 Implementazione

La figura 5 permette di avere uno sguardo d'insieme della struttura del progetto; i segnali con cui questo modulo per la gestione dei convertitori si interfaccia con l'esterno sono:

clk *in*: il master clock del progetto;

rstn *in*: il reset generale del progetto;

dataFromAk *in*: la linea su cui entrano in modo seriale i dati provenienti dal convertitore A/D;

lrck *out*: uno dei clock dell'AK4520A (si veda il par. 2.1);

mclk *out*: uno dei clock dell'AK4520A (si veda il par. 2.1);

sclk *buffer*: uno dei clock dell'AK4520A (si veda il par. 2.1);

dataToAk *out*: la linea su cui escono in modo seriale i dati diretti al convertitore D/A;

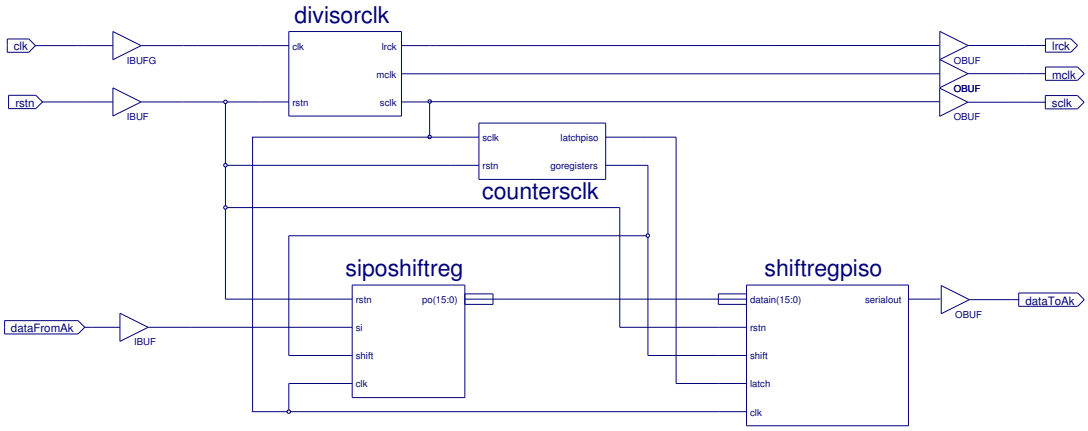


Figura 5: Schema del progetto

Di seguito analizzeremo in dettaglio ogni blocco che compone il progetto.

3.1 `divisorclock`

Compito di questo modulo è di generare i tre clock necessari all'AK4520A: `lrck`, `mclk` e `sclk`. Dal punto di vista dell'implementazione, il modulo altro non fa che gestire un contatore a 10 bit che si incrementa ad ogni colpo di clock; così facendo il bit meno significativo del contatore varierà tra 0 e 1 con una frequenza che è esattamente la metà (2^1) del clock, il secondo bit meno significativo varierà con una frequenza quattro (2^2) volte inferiore a quella del clock e così avanti. Il bit più significativo varierà con una frequenza 1024 (2^{10}) volte inferiore a quella del clock.

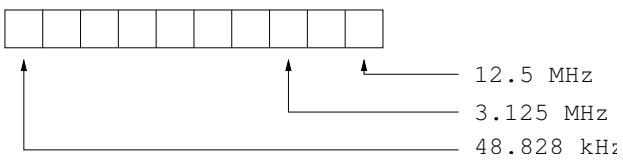
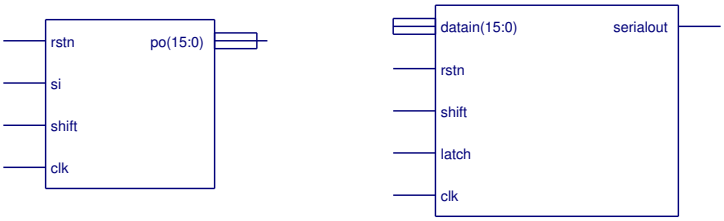


Figura 6: Schema del contatore

Assegnando i segnali `mclk`, `sclk` e `lrck` rispettivamente al primo, terzo e ultimo bit del contatore otteniamo i clock richiesti con uno sforzo minimo.

3.2 `pisoshiftreg` e `shiftregsipo`



Questi due moduli si dedicano a trasformare il segnale digitale da seriale a parallelo e - viceversa - da parallelo a seriale.

si *in*: linea dalla quale entrano nel registro i bit in modo seriale, sincronizzati con `slck`;

shift *in*: quando questa linea è alta i registri lavorano, quando è bassa i registri stanno fermi;

po *out*: bus in cui escono i due byte contenenti un campione;

datain *in*: bus in cui entrano i due byte contenenti un campione;

latch *in*: si veda latchpiso al par. 3.3;

serialout *out*: linea dalla quale escono dal registro i bit in modo seriale, sincronizzati con `slck`.

I due shift-register funzionano in maniera duale: prendiamo come esempio quello Seriale/Parallelo (figura 7): man mano che i bit arrivano in maniera seriale vengono inseriti in un registro a sedici

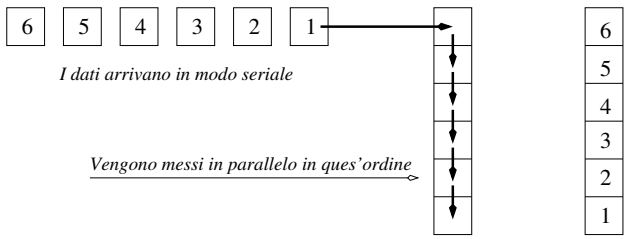


Figura 7: Funzionamento dello shift.register

bit che altro non è che una pila FIFO: quando sono arrivati tutti i bit lo shift-register viene "congelato" e si preleva la pila (ovvero i due byte). C'è da notare che questi registri si trovano sempre in uno stato *coerente*, ovvero è possibile prelevare la pila in qualsiasi momento in quanto essa al sua interno avrà sempre una sequenza di uni e zeri: il controllo viene fatto semplicemente contando i cicli di `sclk` (si veda il paragrafo successivo).

3.3 `countersclk`

Il compito di questo modulo è quello di pilotare i shift-register Seriale/Parallelo e Parallelo/Seriale.

clk *in*: il clock che pilota il modulo: viene collegato non al master clock bensì a sclk;

latchpiso *out*: segnale impulsivo che dice quando bufferizzare il dato all'interno dello shift-register parallelo/seriale (avverte quando il dato è pronto, ovvero quando tutti e 16 i bit sono arrivati dal convertitore A/D e sono stati "parallelizzati"); viene realizzato contando semplicemente i fronti d'onda di sclk a cicli di 32 (tanti quanti ce ne stanno in un semiperiodo di lrck, si veda il par. 2.2) e asserendo la linea quando il contatore assume il valore 17.

goRegisters *out*: sincronizza i registri Seriale/Parallelo e Parallelo/Seriale: fintatto che il segnale è alto gli shift-register prendono in ingresso un segnale seriale (parallelo) e lo "parallelizzano" ("serializzano").

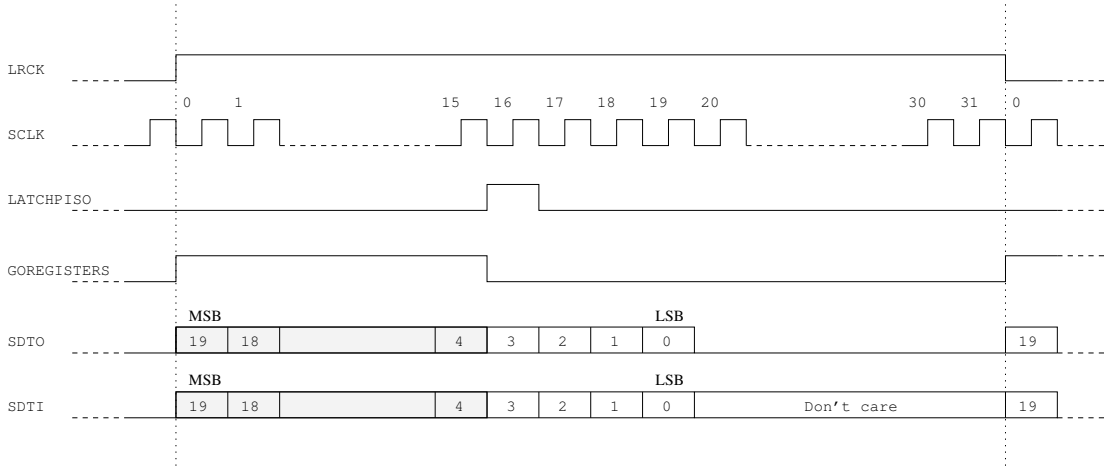


Figura 8: La sincronizzazione degli shift-register

4 Funzionamento

Il funzionamento di questo modulo è assolutamente semplice: basta inserire una sorgente sonora (preamplificata) nel jack IN e collegare un paio di cuffie (o due casse) nel jack OUT. Una volta fatto il download del binario sulla scheda i convertitori iniziano a funzionare pilotati dai clock e con essi i registri Seriale/Parallelo e Parallelo/Seriale. Il tasto di reset è presente per poter resettare manualmente tutti i moduli in qualunque momento: benché in questo contesto non sia strettamente necessario è stato inserito per renderlo disponibile a chi volesse in futuro sviluppare ulteriormente questo progetto. L'implementazione di questi moduli è stata svolta avendo come obiettivo primario l'economia delle (poche) risorse presenti sull'XSA-50: lo scopo è stato quello di scrivere una logica che fosse la più semplice e piccola possibile al fine di occupare il minor numero di risorse. Questo è il motivo per cui spesso si è preferita la semplicità dell'implementazione a discapito - ad esempio - dell'eleganza.

Nota: si faccia attenzione a non avere volumi elevati nelle casse al momento del download sulla scheda o al momento di un reset manuale; in genere le membrane delle casse (o delle cuffie, nonché le orecchie...) sono molto sensibili e mal sopportano i picchi di tensione.

Bibliografia

[**1**] AK4520A data sheet:
http://www.xess.com/manuals/AK4520A.pdf

[**2**] VHDL Reference Manual, Synario:
http://cslab.snu.ac.kr/course/cad99/vhdl_ref.pdf

[**3**] XSA-50 Board V1.2 Manual
http://www.xess.com/manuals/xsa-manual-v1_2.pdf

[**4**] XStend Board V2.0 Manual
http://www.xess.com/manuals/xst-manual-v2_0_0.pdf