Implementazione in VHDL di un

Delay audio in tempo reale

Su schede: Xess XSA-50
Xess XTD-2

Christian Gregorutti

1 Introduzione

Questo progetto si propone di realizzare un delay audio su scheda Xess XSA-50 che funzioni a tempo reale; ovvero (come mostra la figura 1) un modulo che accetti in ingresso un segnale audio e lo riproponga in uscita sommato a infinite sue ripetizioni che si smorzano man mano nel tempo.

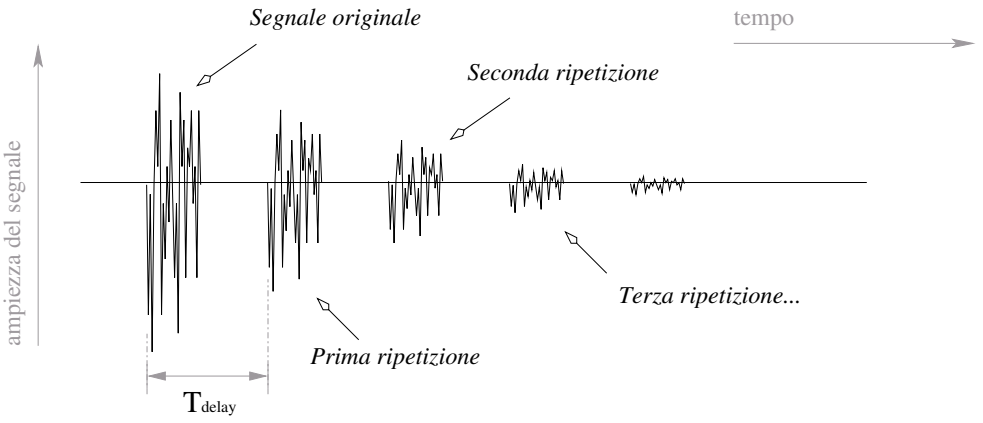

Figura 1: Segnale in uscita

Il progetto lavora con un master clock di 50 MHz e converte il segnale audio con una frequenza di campionamento di 48.828 kHz e 16 bit per campione (qualità CD[1]); accetta in ingresso un qualsiasi segnale audio purché preamplificato: in questo contesto è stata utilizzata l'uscita di una scheda audio da PC, tuttavia è possibile collegare in ingresso anche un microfono a patto di abilitare la preamplificazione del segnale sulla scheda XStend Board V2.0 settando opportunamente i jumper vicino ai convertitori. Questo progetto riutilizza i moduli per la gestione di tastiera, memoria e convertitori audio frutti di un precedente lavoro, la cui documentazione si trova anch'essa all'interno dell'archivio contenente i sorgenti.

2 Funzionamento

La scheda accetta in ingresso un segnale audio stereo dal jack J1, lo elabora e lo rimanda in uscita al jack J2 (dove si possono collegare un paio di cuffie o delle casse attive). Inoltre tramite la tastiera (collegata alla porta PS2 sulla XSA-50) è possibile cambiare il tempo del delay T_{delay} (figura 1) durante il funzionamento.

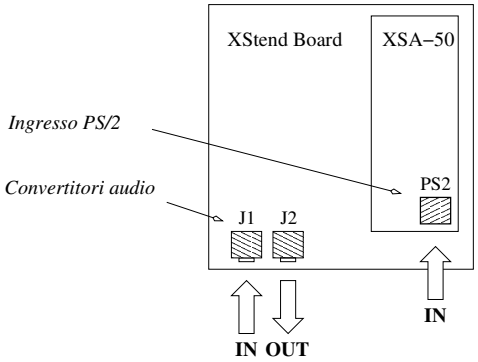

Figura 2 Ingressi e uscite del modulo

Una volta collegata la scheda, non appena il binario viene scaricato sull'FPGA vengono inizializzate tastiera e memoria e i convertitori iniziano a funzionare: di conseguenza il segnale d'ingresso viene elaborato e mandato in uscita; di default all'inizio il valore di T_{delay} è pari a zero (ovvero in uscita è riproposto il segnale in ingresso inalterato). A questo punto bisogna dire alla logica interna all'FPGA di attivare il delay: per fare ciò basta premere il tasto `t` della tastiera e selezionare una cifra (corrispondente a T_{delay} espresso in decimi di secondo) dal tastierino numerico: in uscita si sentirà il segnale originale sommato alle sue ripetizioni.

[1] A essere precisi i cd lavorano a 44.100 kHz

3 L'implementazione

In questo contesto non ci occuperemo della gestione della tastiera, della memoria e dei convertitori in quanto argomenti già trattati precedentemente: è quindi sufficiente ricordare che ogni campione audio prelevato dai convertitori (un campione per canale ogni $48.828.000^{-1}$ secondi) viene codificato con un treno di 16 bit (successivamente messi in parallelo su un bus di 2 byte) e quindi memorizzato in *una* locazione di memoria; in uscita il procedimento è il medesimo.

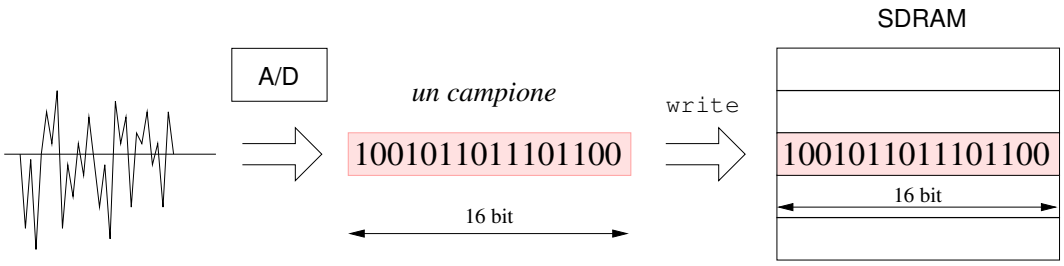

Figura 3: 16 bit per campione

Prima di analizzare in dettaglio la struttura del modulo è necessario soffermarsi un attimo sulla codifica del segnale audio.

3.1 La codifica del segnale

Ogni campione audio viene codificato su 16 bit in complemento a 2; questo significa che il convertitore codifica con 7FFFFH(@20bit)[2] 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). Il complemento a due di un numero positivo è uguale alla sua rappresentazione binaria; per quanto riguarda i numeri negativi (per esempio -7) il complemento a due si ottiene invertendo il modulo del numero bit per bit (7 @4bit = `0111` che invertendo diventa `1000`) e sommando 1 (`1001`). Il punto di forza di questa rappresentazione è la somma algebrica che viene fatta (si veda esempio sottostante) come se i due operandi fossero la rappresentazione binaria di due numeri interi.

```
0000 =  0
0001 =  1
0010 =  2        0101 +       5 +
 ...             1101 =      -3 =
0111 =  7       ------       ---
1000 = -8        0010         2
1001 = -7
 ...
1111 = -1
```

Un po' meno immediata risulta invece la divisione per una potenza di due; in complemento a due per dividere un numero positivo per una potenza di due è sufficiente effettuare uno shift a destra di tanti posti quanto l'esponente del divisore (se dobbiamo dividere per $4 = 2^2$ dobbiamo shiftare di 2 posizioni); quando invece si ha a che fare con un numero negativo si segue la medesima operazione con l'accortezza di replicare il bit del segno per ogni posizione che andremo a scalare: il procedimento è semplice da capire osservando la figura 4.

[2]Ricordiamo che il convertitore codifica a 20 bit di cui noi ne prediamo i 16 più significativi.

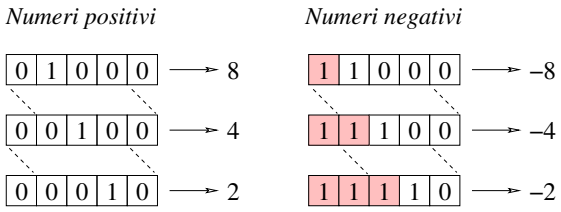

Figura 4: Divisione in complemento a 2

3.2 Il loop

Per poter realizzare un delay audio che proponga in uscita *infinite*[3] ripetizioni del segnale è necessario realizzare un *loop* che prenda il segnale dall'uscita e lo riporti in ingresso attenuato e con un ritardo pari a T_{delay}; abbiamo dunque bisogno di implementare:

il ritardo T_{delay} la cui implementazione viene gestita direttamente dalla memoria (si veda a questo proposito la sezione 3.3);

l'attenuazione del segnale di cui si occupa il modulo *shift* (si veda la figura 5) che in realtà è una semplice riga di codice:

```
dataToSdramLow <= dataToSdram(15) & dataToSdram(15 downto 1);
```

che effettua una traslazione a destra del numero binario (divisione per due) replicando opportunamente il bit di segno (divisione in complemento a due).

Osserviamo che in figura 5 sono presenti due *switch* implementati dalla riga di codice:

```
dataToSdram <= dataFromSipo when timeDelay=0 else dataFromMix;
```

Questo fa in modo che quando il T_{delay} è nullo allora in memoria viene memorizzato il segnale audio direttamente dall'ingresso e viene immediatamente riproposto in uscita senza modifiche. Non appena T_{delay} assume valori diversi da zero il loop si chiude e ci troviamo di fronte a una retroazione del segnale; in memoria troviamo pertanto il segnale originale a cui vengono sommate copie di esso opportunamente traslate e ridotte in ampiezza.

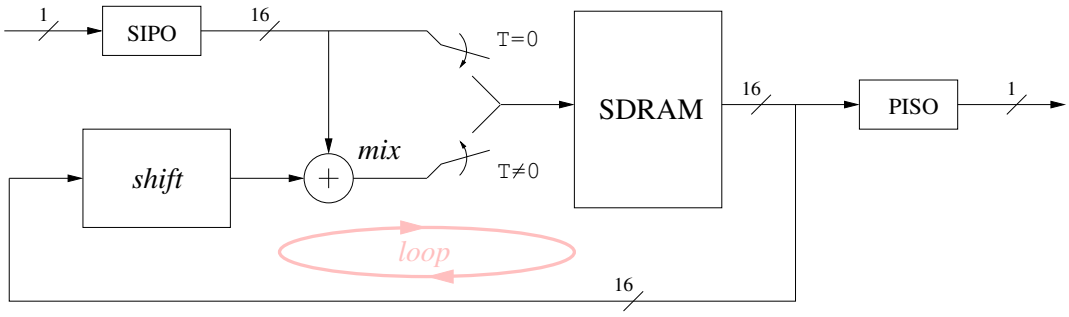

Figura 5: Schema logico del delay

A questo punto possiamo analizzare in dettaglio i singoli moduli che compongono questo lavoro: verranno presi in esame solo quelli che interconnettono tra loro memoria, tastiera e convertitori per realizzare il delay: per la gestione delle singole periferiche si rimanda alla relativa documentazione.

[3] In teoria sono infinite, in pratica dopo circa dieci smorzamenti le ripetizioni diventano impercettibili.

3.3 `addressGenerator`

Questa entity serve a generare gli indirizzi di memoria che vengono inviati al controller della SDRAM: ovvero tramite questo modulo si specifica in quale locazione di memoria effettuare l'operazione corrente di lettura/scrittura. Nel caso in cui il tempo del delay sia impostato a zero (settaggio di default al momento dell'*accensione*) allora a partire dalla locazione 0 man mano che i campioni arrivano vengono memorizzati in memoria e subito "ripescati" e mandati in uscita; quando si raggiunge l'ultima locazione di memoria disponibile si riprende dall'inizio (figura 6-1).

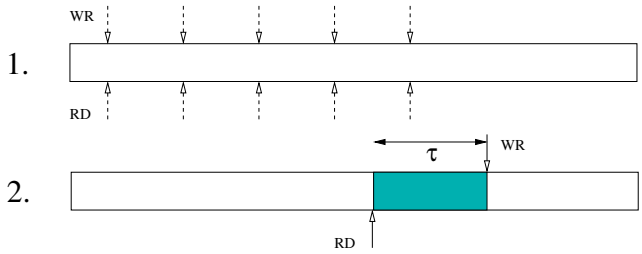

Figura 6: Gestione del ritardo

Questo modulo gestisce due "puntatori": uno (RD) che fa riferimento alla locazione di memoria su cui effettuare la lettura e uno (WR) che fa riferimento alla locazione di memoria su cui effettuare la scrittura (possiamo immaginarli come la testina di lettura e di scrittura di un nastro magnetico). Quando T_{delay} è nullo allora i due puntatori sono allineati (puntano alla stessa cella di memoria), ovvero il campione appena inserito in memoria viene subito letto.

Quando invece viene inserito il delay la testina di lettura si sposta indietro di τ (numero di locazioni necessarie a memorizzare T_{delay} secondi di musica - figura 6.2) rispetto a quella di scrittura: quest'ultima continua a scrivere su memoria il segnale d'ingresso a cui però viene sommato il segnale proveniente dalla testina di lettura (posizionata T_{delay} secondi prima) dimezzato di volume.

Per il calcolo di τ se consideriamo che 1 secondo sono 48828 campioni per canale (in tutto 97656 campioni) allora possiamo stilare la seguente tabella:

```
decimi di secondo => numero di locazioni

1 =>  9766     3 => 29297     5 => 48828     7 => 68359     9 => 87890
2 => 19531     4 => 39062     6 => 58594     8 => 78125
```

3.4 `WriteRead`

Il modulo `WriteRead` si occupa di pilotare il controller della SDRAM dicendogli quando scrivere o leggere in memoria, occupandosi di fornire gli indirizzi e di gestire i dati che entrano in memoria o che da essa escono.

L'implementazione è realizzata con una macchina a stati (figura 7): al momento del reset la macchina passa in un stato di attesa (STOP) e si mette in ascolto della linea `goWR`; non appena questa viene asserita, e per tutto il tempo in cui rimane alta, la macchina a stati prepara sui relativi bus i dati da scrivere in memoria e gli indirizzi. Quando la linea `goWR` ritorna bassa viene mandato al controller della SDRAM l'ordine di effettuare la scrittura. Seguendo il protocollo del controller, tale ordine (`wr=1`) deve rimanere attivo per tutta la durata della scrittura e terminare solo quando il controller informa il resto della logica sull'avvenuta operazione (`done=1`). A questo punto un campione è stato appena memorizzato in memoria. Segue immediatamente l'inizio dell'operazione di lettura mediante le stesse modalità; appena la memoria ha terminato anche l'perazione di lettura la macchina a stati finiti ritorna in stato di attesa.

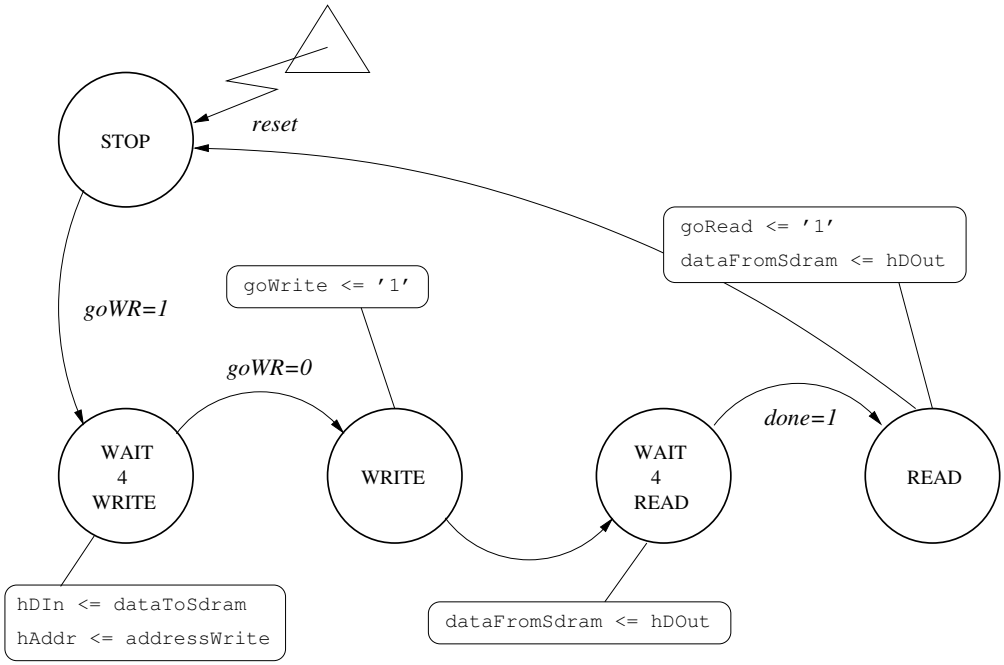

Figura 7: Macchina a stati finiti

Ogni accesso in memoria consiste di una scrittura e da una lettura che avvengono consecutivamente; questa modalità di accesso è dettata dal modo in cui funzionano i convertitori: questi ultimi infatti ogni $(2 * 48828)^{-1}$ secondi ricevono e spediscono un campione; considerando il tempo necessario al modulo seriale-parallelo per convertire il treno di 16 bit in due byte e il tempo necessario al registro parallelo-seriale per bufferizzare i due byte in uscita ci accorgiamo di come la memoria abbia una finestra temporale di 11 cicli di `sclk`[4] per memorizzare un campione, prelevare il campione τ locazioni precedenti, effettuare la somma tra i due e presentare il risultato in uscita.

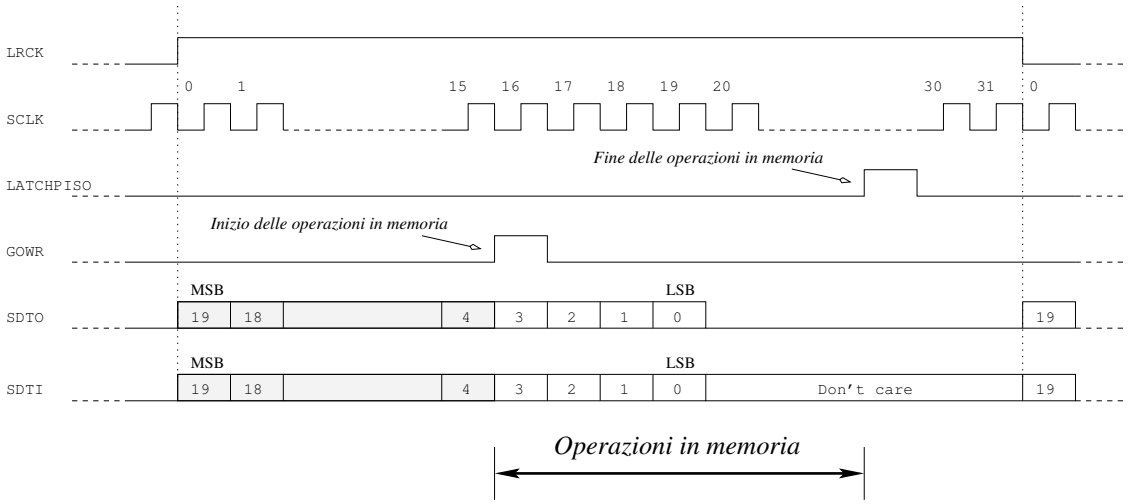

Figura 8: Temporizzazioni della memoria

[4]Visti i tempi con cui lavora la SDRAM questa finestra temporale è un vincolo tutt'altro che stretto.

Bibliografia

[**1**] Modulo per la gestione della memoria SDRAM
./sdram.pdf

[**2**] Modulo per la gestione dei convertitori audio
./audio.pdf

[**3**] Modulo per la gestione della tastiera
./keyboard.pdf

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

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

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