# ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

### **SCUOLA DI INGEGNERIA E ARCHITETTURA**

### Dipartimento di Informatica – Scienza e Ingegneria

### Corso di Laurea Magistrale in Ingegneria Informatica

### **Progetto finale ed attività progettuale**

### in

### SISTEMI DIGITALI M

# Briscola con Altera DE1 Board



CANDIDATI:

|  |  |  |
| --- | --- | --- |
| Lorenzo Mustich | Alessandro Paoletti | Alessandro Morabito |
| 0000901917 | 0000901759 | 0000907048 |

Anno Accademico: 2019/2020

# Sommario (da inserire solo alla fine)

[Capitolo 1. La Briscola 1](#_Toc18877272)

[Capitolo 2. Architettura del sistema 2](#_Toc18877273)

[Capitolo 3. Datapath 3](#_Toc18877274)

[Capitolo 4. Control Unit 4](#_Toc18877275)

[Capitolo 5. Graphic User Interface: il lato Java 5](#_Toc18877276)

[Capitolo 6. Punto d’incontro: comunicazione seriale 6](#_Toc18877277)

[Capitolo 7. Audio 7](#_Toc18877278)

[7.1 Codec stereo WM8731 7](#_Toc18877279)

[7.2 Protocollo I2C 8](#_Toc18877280)

[7.3 Interfaccia PCM 13](#_Toc18877281)

[7.4 Entità di top-level 14](#_Toc18877282)

[Capitolo 8. Fonti 15](#_Toc18877283)

# Capitolo 1. La Briscola

## 1.1 ciao

# Capitolo 2. Architettura del sistema

# Capitolo 3. Datapath

# Capitolo 4. Control Unit

# Capitolo 5. Graphic User Interface: il lato Java

# Capitolo 6. Punto d’incontro: comunicazione seriale

# Capitolo 7. Audio

Per sfruttare al meglio le potenzialità della scheda, abbiamo deciso di inserire un sottofondo musicale che è possibile ascoltare durante la partita.

## 7.1 Codec stereo WM8731

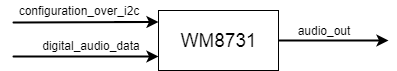
La DE1 contiene un codec stereo appositamente progettato per la lettura di file mp3, il WM8731.

Figura 1. Schema funzionamento codec audio

Come si evince dalla Figura 1, il codec legge dei dati audio digitali sotto forma di stringhe da 16 a 32 bit, campionandole con una frequenza variabile da un minimo di 8KHz ad un massimo di 96 KHz. Può funzionare sia come dispositivo MASTER che come SLAVE. Possiede un’interfaccia che fornisce:

* controlli di volume;
* possibilità di silenziare l’audio;
* possibilità di usare il codec in modalità stereo o mono

il tutto configurabile tramite il protocollo I2C.

La configurazione da noi usata per questo progetto è la seguente:

* codec in SLAVE mode;
* campionamento in USB mode:
  + clock a 12 MHz;
  + frequenza di campionamento di 48kSps;
  + risoluzione audio da 16 bit
* campionamento tramite PCM (*pulse-code modulation*)

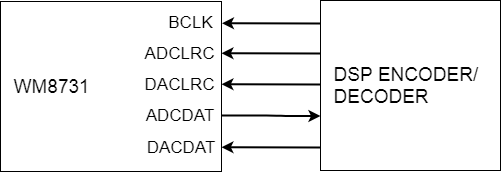
Il DSP (*Digital signal processor*), mostrato in figura 2, è un hardware dedicato e ottimizzato presente sulla scheda DE1 che permette di elaborare efficientemente segnali digitali.

Figura 2. WM8731 in SLAVE mode

Il WM8731, in SLAVE mode, riceve dal DSP il clock a cui tutto il sistema audio lavorerà (*BCLK, Digital Audio Bit Clock*) e i due clock per la sincronizzazione delle operazioni di conversione da analogico a digitale (*ADCLRC, ADC sample rate left/right clock*) e viceversa (*DACLRC, DAC sample rate left/right clock*); dopodichè, il codec stereo invia i dati audio da convertire (*pin* *ADCDAT, ADC digital audio data output*) e ne riceve la conversione (*pin* *DACDAT, DAC digital audio data output*).

## 7.2 Protocollo I2C

L’I2C (*Inter Integrated Circuit*), sviluppato nel 1982 dalla Philips, ma divenuto uno standard largamente diffuso solo nel 1992, è il protocollo che abbiamo scelto per la configurazione del codec audio.

Trattasi di un protocollo seriale che permette l’interfacciamento tra uno o più device *master* con un numero illimitato di altri device *slave*. Sono presenti solo due bus:

* SCL: *serial clock*, utilizzato per definire un clock unico per la connessione;
* SDA: *serial data*, per i dati effettivi;

entrambi i bus sono bidirezionali.

I device *master* generano il clock, danno inizio e fermano la connessione con i vari altri componenti e inviano i vari comandi permessi dal protocollo.

Una trasmissione dati base avviene in questo modo:

1. in principio, i due bus sono posti a livello alto;
2. il *master* genera una *Start condition* (SCL = 1, SDA = 0) seguito dall’indirizzo dello *slave* interessato: se il LSB è posto a 0, il master intende scrivere (*write mode*);
3. attende un *ack* da parte del device scelto;
4. in caso di *ack* positivo, invia un byte di dato;
5. attende un *ack* da parte del device scelto;
6. i punti 5 e 6 si ripetono fintanto che non venga generata una *Stop condition* (SCL = 1, SDA = 1).

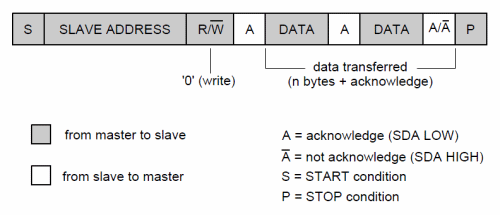
Di seguito è mostrato tramite l’uso di diagrammi temporali quanto appena spiegato.

Figura 4. Esempio di comunicazione I2C

Ai fini dell’implementazione di *BriscolaDE1*, è stata definita un’*entità* apposita che implementasse il protocollo scelto.

Figura 3. Funzionamento del protocollo I2C

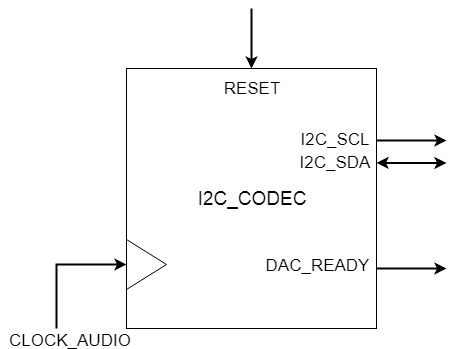
Il suo comportamento è stato definito utilizzando i seguenti *process*:

Figura 5. Entità I2C

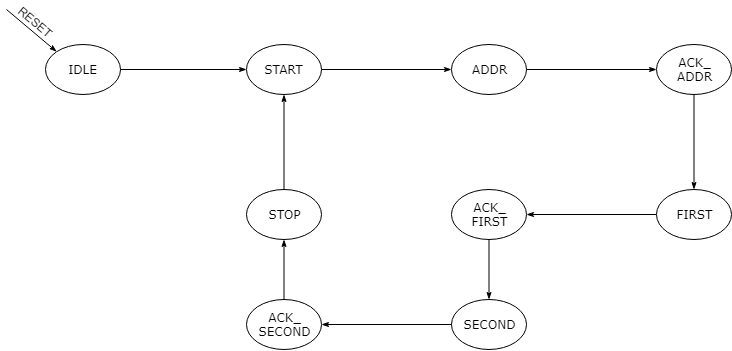
* I2CClockProcess : process(CLOCK\_12, RESET)
  + si occupa di generare, tramite l’utilizzo di un prescaler, i clock utili al protocollo e le transizioni tra i dati;
* FSMI2CProcess : process(CLOCK\_12, RESET)
  + definisce un automa a stati finiti che scandisce il regolare funzionamento del protocollo; a grandi linee, può essere visto nel seguente modo:

Figura 6. ASF dell’I2C\_codec

* InitProcess : process(CLOCK\_12, RESET)
  + si occupa della configurazione vera e propria del codec WM8731

case init\_counter is

when 0 =>

-- reset

i2c\_data(15 downto 9) <= "0001111";

i2c\_data(8 downto 0) <= "000000000";

i2c\_send\_flag <= '1';

when 1 =>

-- active interface

i2c\_data(15 downto 9) <= "0001001";

i2c\_data(8 downto 0) <= "111111111";

i2c\_send\_flag <= '1';

when 2 =>

-- ADC off, DAC on, Linout ON, Power ON

i2c\_data(15 downto 9) <= "0000110";

i2c\_data(8 downto 0) <= "000000111";

i2c\_send\_flag <= '1';

when 3 =>

-- Digital Interface: DSP, 16 bit, slave mode

i2c\_data(15 downto 9) <= "0000111";

i2c\_data(8 downto 0) <= "000010011";

i2c\_send\_flag <= '1';

when 4 =>

-- headphone volume

i2c\_data(15 downto 9) <= "0000010";

i2c\_data(8 downto 0) <= "101111001";

i2c\_send\_flag <= '1';

when 5 =>

-- USB mode

i2c\_data(15 downto 9) <= "0001000";

i2c\_data(8 downto 0) <= "000000001";

i2c\_send\_flag <= '1';

when 6 =>

-- enable DAC to LINOUT

i2c\_data(15 downto 9) <= "0000100";

i2c\_data(8 downto 0) <= "000010010";

i2c\_send\_flag <= '1';

when 7 =>

-- remove mute DAC

i2c\_data(15 downto 9) <= "0000101";

i2c\_data(8 downto 0) <= "000000000";

i2c\_send\_flag <= '1';

init <= '1';

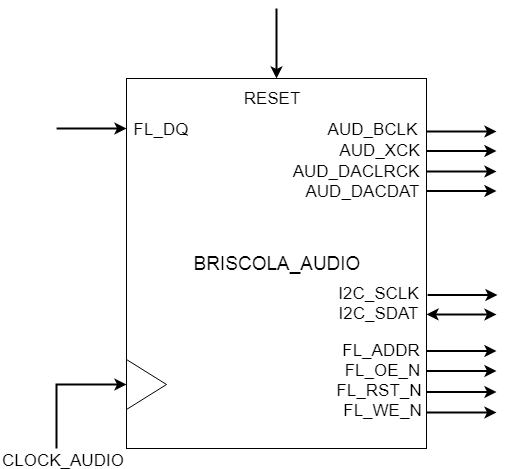
dac\_ready <= '1';

when others => … -- reset data and flag

end case;

## 7.3 Interfaccia PCM

## 7.4 Entità di top-level

In base a quanto mostrato fino a questo punto, l’entità di top-level che si occuperà della riproduzione dell’audio sarà la seguente:

# Capitolo 8. Fonti