UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

RELAZIONE PROGETTO DI ELETTRONICA

di

Kasibovic Amar

CDL: Ingegneria dell'Informazione e delle Comunicazioni

Titolo: **Visualizzazione della trasformata di Fourier di dati audio tramite l'utilizzo della Zedboard**

UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

Obiettivo:

Realizzare un sistema in grado di visualizzare la trasformata di Fourier di un segnale audio fornito in ingresso.

Procedimento e descrizione:

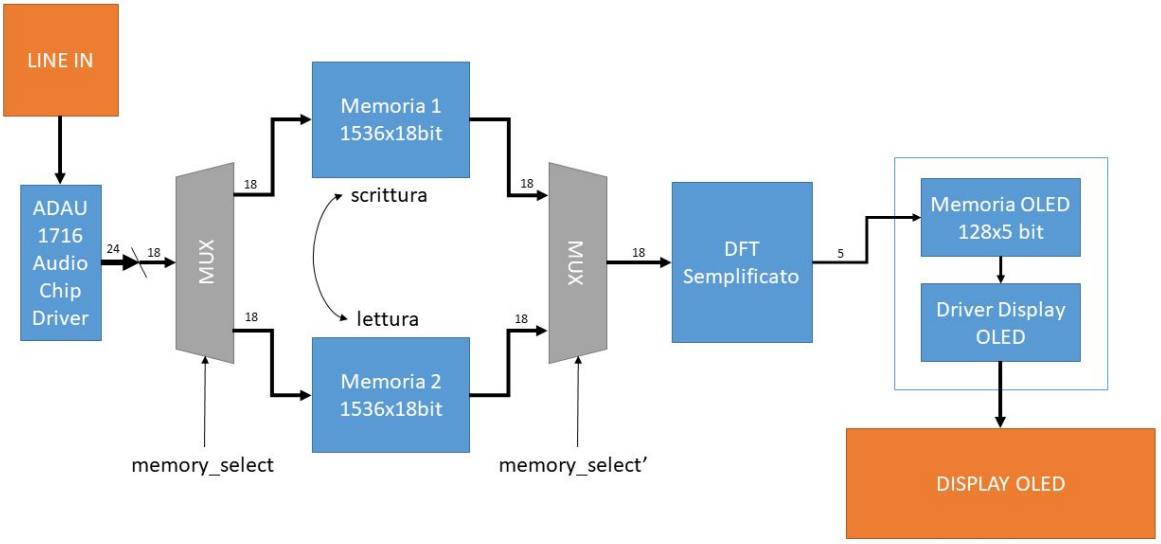
Il sistema è stato realizzato completamente utilizzando il linguaggio VHDL, ed utilizza la sola logica programmabile del chip Zynq presente sulla Zedboard (non è quindi presente software).
Il sistema è composto da vari blocchi: non tutti i blocchi sono stati implementati durante il progetto, bensì si è usufruito di alcuni IP Cores della Xilinx quali il "Memory block generator" per l'utilizzo delle memorie RAM della FPGA, e il "Discrete Fourier Transform" per il calcolo della trasformata di Fourier.
Inoltre per il driver del chip audio (ADAU1716) e del display OLED sono stati utilizzati alcuni codici reperiti in rete, i quali sono stati studiati ed in seguito profondamente modificati (nel caso del driver del display OLED) per adattarli all'utilizzo in questo progetto. In allegato al progetto verranno inseriti i file originali che sono stati modificati.

Data Path:

I dati registrati dal microfono vengono inizialmente forniti alla FPGA tramite il chip audio dedicato, l'ADAU1761, tramite il protocollo I2S.
Il chip audio campiona il segnale audio in ingresso a 48kHz, fornendo quindi 48000 campioni da 24 bit al secondo alla FPGA.
I dati a 24 bit in ingresso vengono troncati al 18esimo bit, e vengono immagazzinati su una tra due memorie RAM, in grado di contenere 1536 campioni da 18 bit ciascuna. Sono utilizzate in modo alternato 2 memorie in quanto, durante l'immagazzinamento dei dati dal microfono su una memoria, i dati vengono prelevati ed elaborati dall'altra memoria.
I 1536 campioni vengono inviati al blocco che implementa la DFT e si attende che tale blocco elabori i dati e fornisca il risultato in uscita. Il blocco DFT supporta una precisione massima di 18 bit sui dati in ingresso, ed è il motivo per cui i campioni in ingresso vengono troncati. Inoltre sono utilizzati 1536 campioni in quanto è un valore adeguato alle nostre esigenze, infatti, elaborando 1536 campioni alla volta, si costruirà uno spettro del segnale in ingresso circa 31.25 volte al secondo: tale frequenza corrisponderà alla frequenza di aggiornamento del display OLED, ed è quindi adatta ad una visione ottimale.
L'uscita della DFT è fornita in un insieme di valori complessi, espressi in parte reale e parte immaginaria, che rappresentano lo spettro del segnale in ingresso. Essendo di nostro interesse la visualizzazione del solo modulo del segnale, è stato aggiunto un blocco in grado di calcolare il modulo a partire dal numero complesso espresso in parte reale e parte

immaginaria. La fase non è stata considerata in quanto non verrà rappresentata sul display OLED. Infine, è stata introdotta un'ulteriore semplificazione dei dati in uscita, in quanto il modulo è stato troncato per ottenere valori a 5 bit: il modulo in tal modo definito, può assumere 2^5 valori differenti, che ben si adattano alla visualizzazione sul display OLED che presenta 32 righe di pixel.
In uscita dal blocco DFT, con 1536 valori in ingresso, si ottengono 1536 valori in uscita. Non è possibile rappresentare tutti i valori sul display, che presenta solo 128 colonne di pixel: i dati in uscita sono quindi confrontati in gruppi da 12, tra i quali viene scelto il valore maggiore che verrà poi visualizzato, ottenendo esattamente 128 valori da rappresentare (1536/12 = 128).
I 128 valori da 5 bit che sono forniti in uscita sono quindi salvati su una memoria RAM, e verranno utilizzati dal driver del display OLED per essere visualizzati.

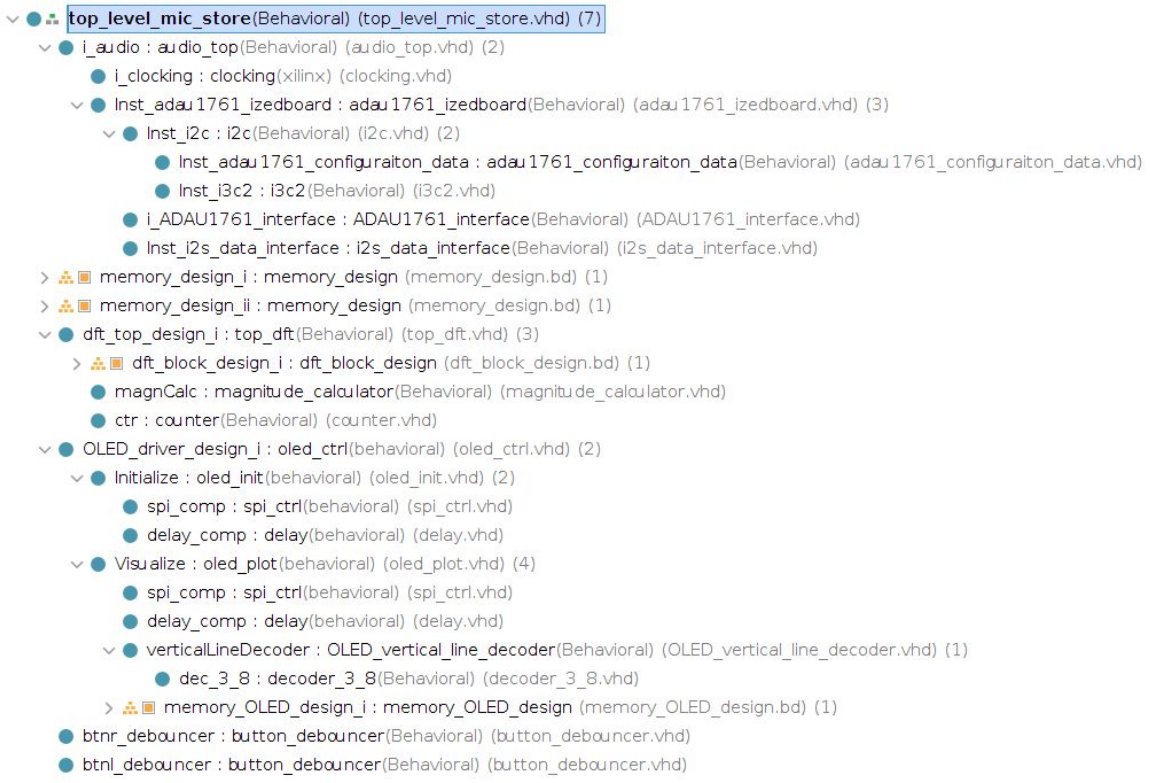
Schema Data Path (semplificato):



UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

Struttura gerarchica dei vari blocchi:



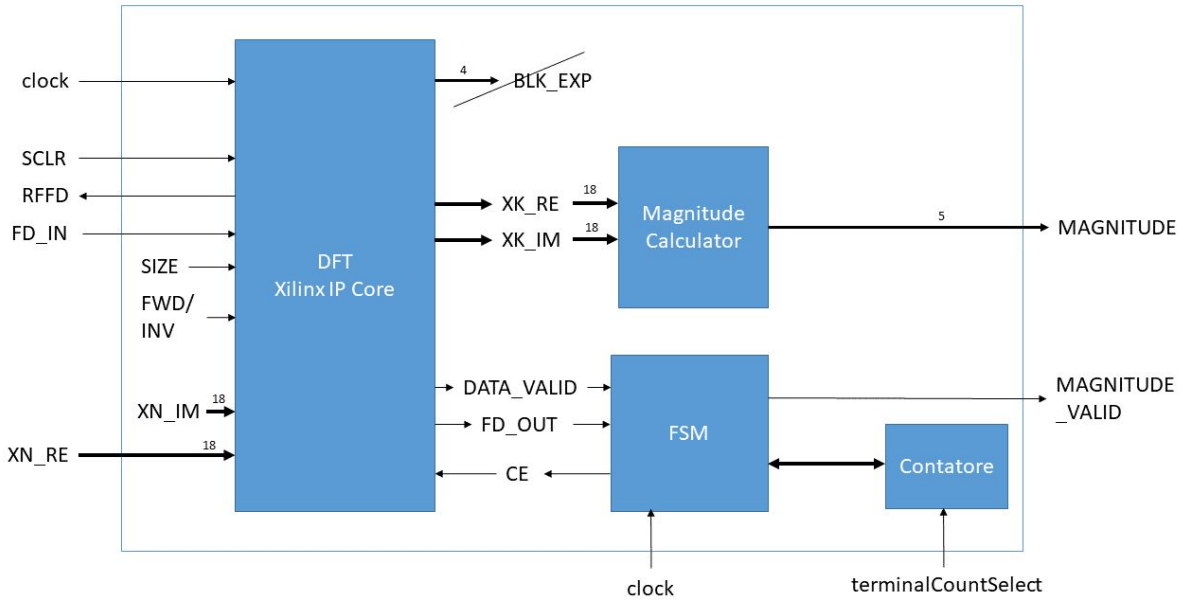
UNIVERSITÀ DEGLI STUDI
DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

BLOCCO DFT SEMPLIFICATO

Il blocco DFT semplificato contiene una istanza dell'IP Core della Xilinx nominato "Discrete Fourier Transform", e implementa una sua semplificazione, tramite il mantenimento di alcuni segnali di ingresso ad un valore costante e la modifica della modalità di rappresentazione dei dati in uscita.

Schema del blocco DFT semplificato:



DFT Xilinx IP Core:

Il blocco della Xilinx è descritto nel seguente documento: DFT Xilinx IP Core Product Guide

Le porte del blocco DFT della Xilinx sono:

XN_RE : (I) Parte reale del segnale in ingresso fornita a 18 bit.
XN_IM : (I) Parte immaginaria del segnale in ingresso. ***Nel blocco semplificato sarà impostato costantemente a 0, in quanto il segnale in ingresso è un segnale reale.***
FD_IN : (I) "First data In", segnala che il primo dato in input è fornito nel ciclo di clock in cui il segnale è alto.
RFFD : (O) "Ready for the first data", segnala all'esterno che il blocco DFT è pronto per ricevere i dati.
SIZE : (I) Dimensione del frame di dati che verrà fornito in ingresso. ***Nel blocco semplificato sarà impostato costantemente a 35, con cui si specifica che si lavorerà su frame da 1536 campioni.***
FWD_INV : (I) "Forward/Inverse", segnala l'utilizzo della trasformata/antitrasformata. ***Nel blocco semplificato sarà impostato costantemente a 1, in quanto è effettuata la sola trasformata.***
SCLR : (I) "Synchronous clear"
CLK : (I) Clock
CE : (I) "Clock Enable", permette di bloccare l'unità, disabilitando il clock.
XK_RE : (O) Parte reale del segnale in uscita.
XK_IM : (O) Parte immaginaria del segnale in uscita.
BLK_EXP : (O) Esponente dei dati in uscita in questo frame di dati. ***Nel blocco semplificato il segnale non verrà considerato, in quanto si vuole rappresentare solo la "mantissa" dei valori in uscita.***
FD_OUT : (O) "First Data Out", segnala che il primo dato è in uscita dal blocco.
DATA_VALID : (O) Segnala che i dati in uscita sono validi.

Magnitude Calculator:

Costituisce un blocco combinatorio, il quale prende i valori di uscita della trasformata di Fourier, espressi con due valori a 18 bit che rappresentano la parte reale e la parte immaginaria, e calcola il modulo del numero complesso espresso in forma polare applicando la seguente formula:

$$MAGNITUDE = \sqrt{XK\_RE^2 + XK\_IM^2}$$

Il calcolo è effettuato nel seguente modo: viene effettuato il quadrato delle due parti reale e immaginaria creando due valori in uscita a 36 bit, vengono poi sommati tali valori ed è calcolata la radice quadrata con l'ausilio di una funzione che implementa un "Non-Restoring Square Root algorithm", il quale fornisce un output di tipo *unsigned* a 16 bit. Per la rappresentazione sul display OLED, verranno considerati solo 5 bit dell'output che costituiranno il valore di "MAGNITUDE".

***Nota**: Si potrebbe pensare che sia utile considerare i 5 bit più significativi dal valore calcolato, ossia dal 15esimo al 11esimo, tuttavia, è stato osservato che nella grande maggioranza dei casi, l'uscita è costituita da valori in cui i 2 bit più significativi sono a 0: per tale motivo vengono forniti in uscita i bit dal 13esimo al 9°.*

E' stato osservato che il blocco combinatorio appena descritto impiega in genere 40ns per stabilizzare l'uscita a partire dall'istante in cui gli ingressi vengono cambiati. Tuttavia, essendo considerati in uscita solo i valori dal bit 9 al 13, il valore si stabilizza prima, creando una **latenza complessiva massima di circa 20ns** su magnitude. Tale latenza deve essere gestita ad un livello superiore.

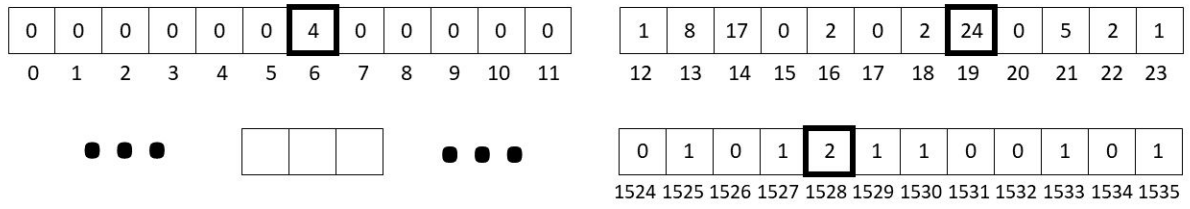
Counter:

Costituisce la semplice implementazione di un contatore a modulo variabile.
L'implementazione è simile a quella classica di un contatore a modulo fisso, ossia presenta un clock di ingresso, un reset sincrono, un count enable, un terminal count (variabile).

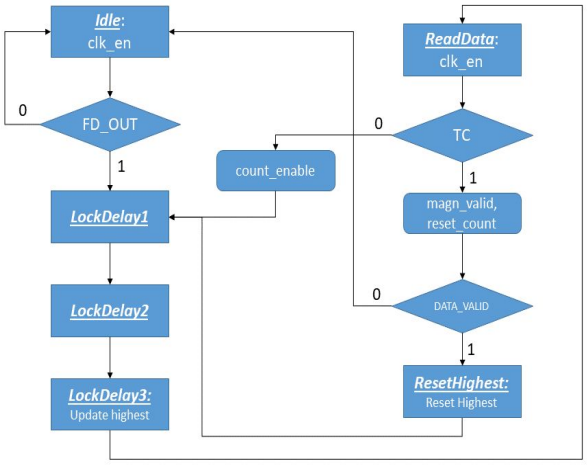
Macchina a stati finiti:

La FSM gestisce i tre blocchi precedenti, in particolar modo essa si occupa della gestione dei dati in uscita dal blocco DFT e la loro semplificazione.
Il blocco DFT fornisce in uscita i valori della trasformata in modo sequenziale, fornendo un nuovo valore ad ogni ciclo di clock. In tal modo però, non si permette al "Magnitude calculator" di effettuare il calcolo che deve effettuare: è necessario introdurre un delay per adattare il sistema alla latenza del magnitude calculator. Durante tale delay, è necessario bloccare il blocco DFT della Xilinx tramite l'abbassamento del clock_enable, affinché non fornisca nuovi dati in uscita, che altrimenti andrebbero persi.
Inoltre, il blocco DFT fornisce in uscita 1536 valori della trasformata, i quali non sono rappresentabili contemporaneamente sul display OLED che presenta solo 128 colonne di pixel: è necessario fornire in uscita solo 128 valori della trasformata invece di 1536. Questo processo è svolto tramite il confronto di gruppi di valori in uscita dal blocco DFT Xilinx: ad esempio, vengono confrontati 1/2/5/12 valori in uscita alla volta, e viene fornito in output solo **il maggiore** di tali valori, fino a quando non sarà raggiunto un totale di 128 valori mandati in output che verranno rappresentati sul display.

In figura è rappresentato un esempio della selezione del valore da mandare in uscita: viene scelto il maggiore tra 12 valori del modulo della trasformata.



La FSM è strutturata nel seguente modo (diagramma di flusso semplificato):



- Quando si ha un dato in ingresso, si blocca l'unità DFT della Xilinx, disabilitando il clock_enable
- Si effettuano due cicli di clock di attesa affinché il calcolo della magnitude_out si stabilizzi
- Si aggiorna il valore maggiore che è stato fornito in uscita nell'ultimo gruppo di valori da confrontare
- Si abilita l'IP Core DFT per un ciclo di clock, si ripete il delay e l'aggiornamento del valore maggiore varie volte.
- Se il gruppo di valori da confrontare è finito, il valore maggiore viene resettato.
- Se i dati in uscita dal blocco DFT Xilinx sono finiti, si va nello stato di attesa.

UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

DRIVER DISPLAY OLED

Il driver è stato reperito in rete. Tuttavia sono state apportate pesanti modifiche per adattare il suo utilizzo alle esigenze del progetto.

Sono stati rimossi: l'ottimizzazione per la visualizzazione a schermo dei caratteri, la libreria ASCII implementata su una ROM che si occupava della conversione dei codici ascii in bit map, la visualizzazione delle schermate di hello world e dell'alfabeto.

Sono stati aggiunti: la possibilità di visualizzare un set di 128 valori a 5 bit sotto forma di istogramma (è stata aggiunta una memoria per salvare e poi leggere tali valori), la visualizzazione di 4 possibili schermate iniziali che segnalano il range di frequenze che si visualizza sul display

Il file top level implementa una macchina a stati che esegue la sequenza di inizializzazione del display, tramite l'utilizzo dell'entità oled_init, ed in seguito si occupa della visualizzazione della trasformata sul display, tramite l'utilizzo di oled_plot. Per trasmettere i dati al display è utilizzato il protocollo SPI, in cui i dati sono inviati in modo seriale.

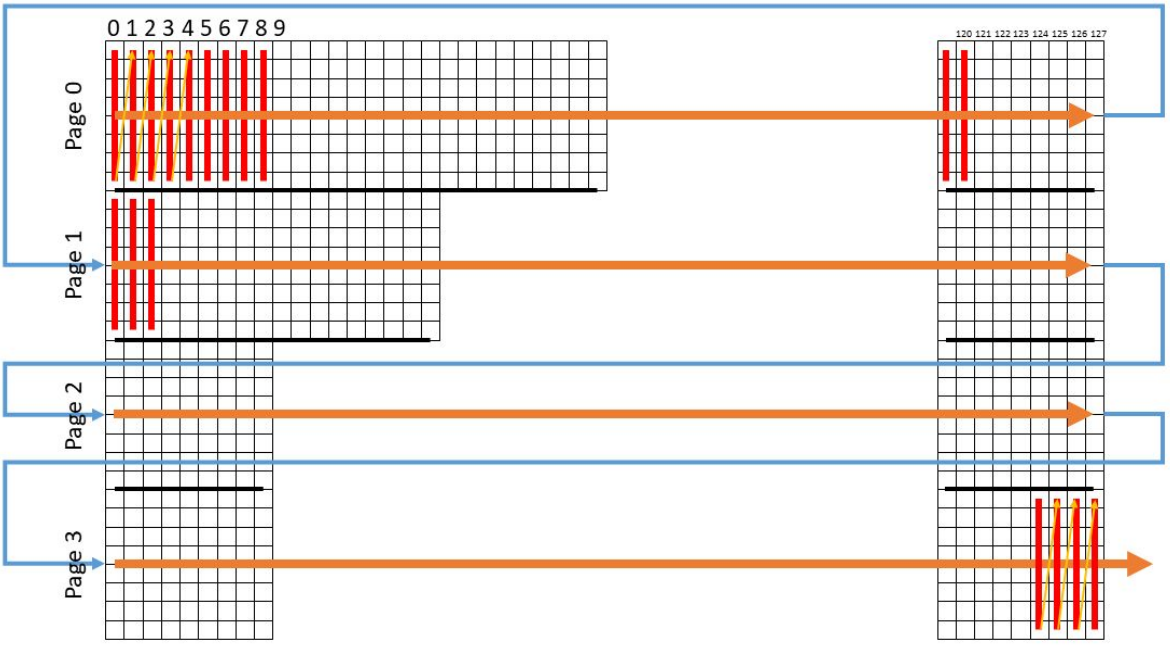
Oled init:

Contiene una macchina a stati che effettua una sequenza di operazioni di inizializzazione del display OLED. Viene usato al suo interno il protocollo SPI per l'invio dei dati al chip del display, e viene istanziato un timer (delay), per temporizzare alcune operazioni di inizializzazione.

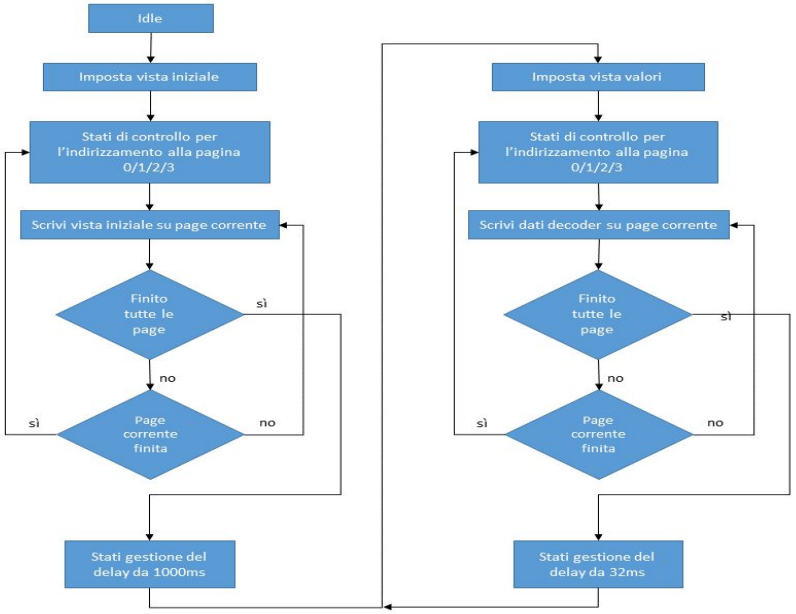
Oled plot:

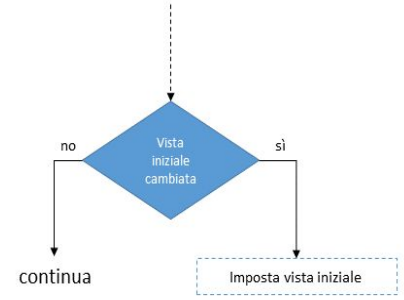
Viene implementata nel file una macchina a stati che si occupa della visualizzazione del set di valori che rappresentano la trasformata.

I valori da rappresentare sul display sono 128 x 5 bit.
Per rappresentarli, bisogna convertire l'informazione contenuta nei 5 bit, in 32 valori di "acceso/spento" da inviare sui led per rappresentare la singola barra dell'istogramma. Per tale scopo è stato utilizzato un decoder di linea verticale.
A complicare la questione vi è inoltre la modalità con cui vengono effettuate le scritture sullo schermo: i 128x32 pixel sono divisi in 4 righe da 8 pixel di altezza (chiamate in inglese *"page"*), la scrittura avviene una page dopo l'altra, e i dati vengono inviati colonna per colonna (8 bit alla volta)(le colonne sono chiamate *"segmenti"*). Gli 8 bit sono mandati uno ad uno, ossia pixel per pixel, in modo seriale utilizzando il **protocollo SPI**.
In figura è rappresentata la modalità con cui vengono inviati i dati da rappresentare al display:



Si comprende quindi come la rappresentazione di barre verticali, che spaziano tra tutte e 4 le page, necessiti di un **decoder** in grado di riconoscere in base al valore a 5 bit in ingresso, quali pixel deve accendere tra i vari gruppi da 8 a seconda del segmento e della page.
Il decoder è descritto nei file *OLED_vertical_line_decoder*, e *decoder_3_8*.
Inoltre, per poter selezionare la page su cui scrivere, è necessaria una sequenza di operazioni di controllo da inviare al display (come l'impostazione dell'indirizzamento "a pagine" e la selezione della page su cui si vuole scrivere).
Tutta questa serie di operazioni sono gestite da una macchina a stati finiti, *che è rappresentata in figura (diagramma di flusso semplificato)*:





<u>Nota</u>: la condizione rappresentata qui in alto a destra è valida durante ogni stato in modo incondizionato.

I valori da consegnare al decoder sono letti dalla **memoria** che è istanziata all'interno di oled_plot. La scrittura su tale memoria è gestita dell'unità DFT semplificata.

UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

DRIVER AUDIO

Il driver audio è un'unità estremamente complessa, per tale motivo, non è stata implementata durante il progetto, ma è stata reperita sul web e adattata con piccole modifiche alle esigenze del progetto.

Descrizione driver audio originale
Repository driver audio originale

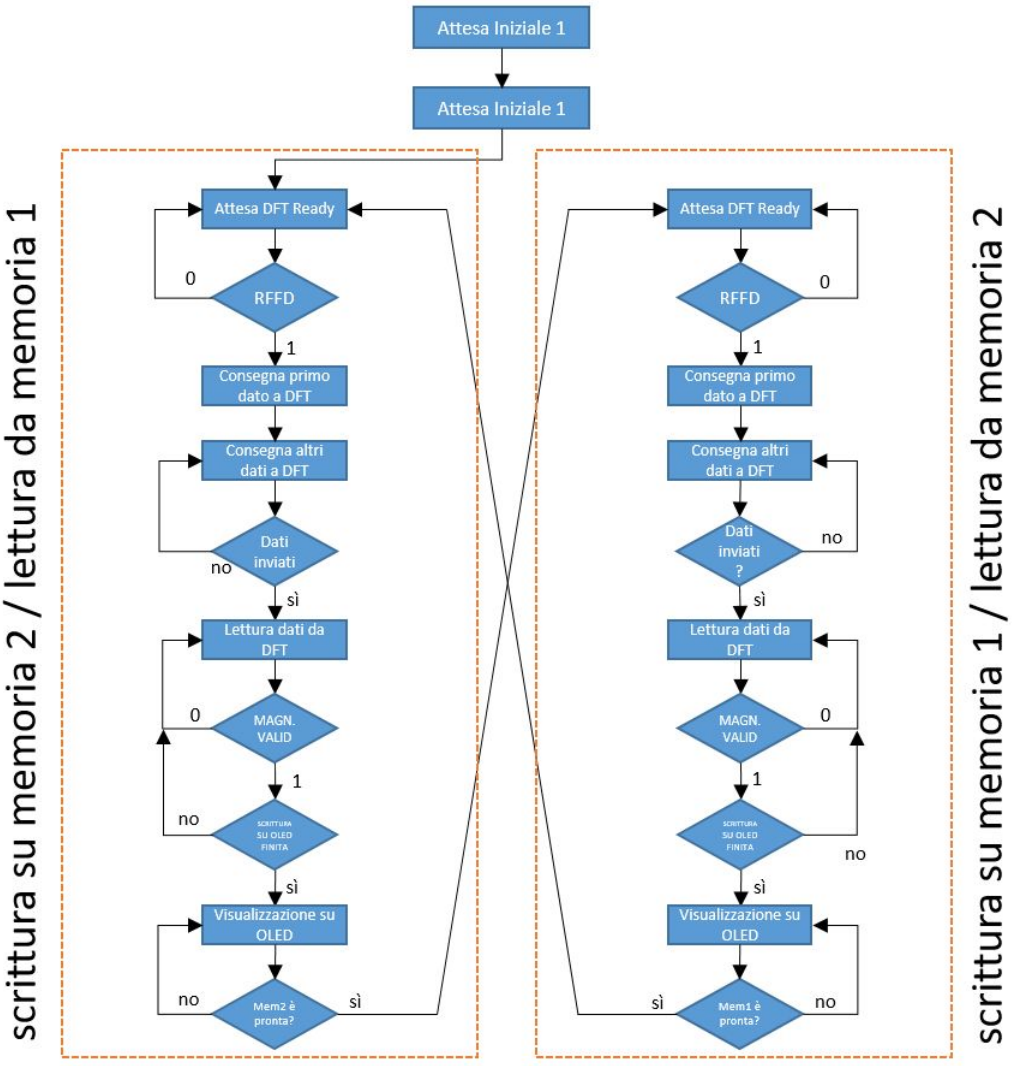
Un prossimo obiettivo potrebbe essere studiare il data sheet del chip ADAU1761 ed implementare un proprio driver (data sheet ADAU1761).

ENTITÀ TOP LEVEL

Il file top level contiene le istanziazioni di tutti i blocchi precedentemente descritti e rappresentati nello schema del data path semplificato precedentemente mostrato in figura.
Le istanze presenti sono:

- driver audio
- 2 memorie 1536 x 18 bit per il salvataggio dei dati registrati da microfono
- blocco dft semplificato
- driver display oled
- (2 debouncer per l'utilizzo dei pulsanti fisici)

Per gestire il salvataggio dei dati ricevuti dal microfono attraverso il driver audio è stato utilizzato un processo (*scritturaDaMic*), che in funzione della memoria selezionata per la scrittura (scelta tramite *memory_select*) scrive i dati su una delle due memorie.
La memoria da cui i dati vengono letti per l'elaborazione è selezionata tramite il negato di *memory_select*, e l'elaborazione e visualizzazione dei dati è gestita tramite una macchina a stati finiti, *il cui schema è qui illustrato in modo semplificato:*



La macchina a stati può essere spiegata nel seguente modo:

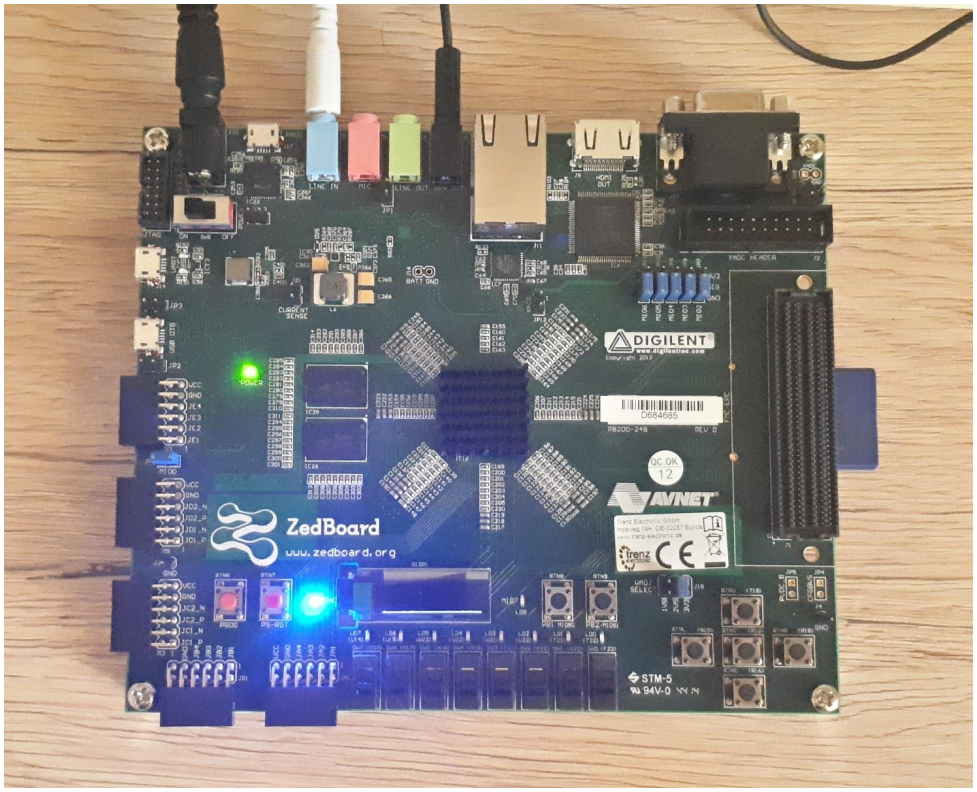
1. attendo che la DFT sia pronta, ossia quando essa imposta RFFD (ready for the first data) alto
2. invio il primo dato segnalando che si tratta del primo tramite l'invio del segnale FD_IN = '1' (first data in) per un ciclo di clock
3. invio tutti gli altri dati
4. attendo che il blocco DFT mi invii i valori della trasformata e li salvo quando arrivano
5. quando ho salvato tutti i valori li visualizzo sul display OLED e attendo che l'altra memoria su cui sto scrivendo i dati dal microfono sia stata riempita
6. ripeto i passaggi da 1 a 5 scambiando le memorie dove sto salvando i dati del microfono (inverto le funzionalità di lettura e scrittura)

UNIVERSITÀ DEGLI STUDI
DI TRENTO

**Dipartimento di Ingegneria e Scienza
dell'Informazione**

Conclusioni e modalità di utilizzo della scheda

I diversi file permettono di generare un bitstream che può essere utilizzato per programmare la Zedboard. I jumper della zedboard devono essere impostati nella modalità di programmazione tramite USB Jtag (ossia i jumper JP8, JP9, JP10 devono essere connessi a GND, mentre i restanti jumper possono rimanere nella posizione di default).



Una volta programmata la zedboard, sarà possibile collegare alla porta "LINE IN" della Zedboard un microfono (o in alternativa un cavo aux collegato all'uscita di una scheda audio, come ad esempio l'uscita delle cuffie del pc). E' possibile collegare anche delle cuffie o una cassa sulla porta "HPH OUT" tramite un cavo aux per ascoltare ciò che è fornito in ingresso alla Zedboard.
Una volta collegate le suddette periferiche, verrà visualizzata sul display la trasformata di fourier del segnale fornito su LINE IN.
E' possibile visualizzare vari range di frequenze: 0-4kHz, 0-8kHz, 0-20kHz o 0-48kHz. Si può navigare tra i diversi range utilizzando i pulsanti destro e sinistro presenti fisicamente sulla zedboard.
E' utile resettare in alcuni casi il display tramite il pulsante centrale, in quanto è possibile che vi siano degli errori nell'indirizzamento dei pixel del display che fanno sì che l'immagine venga traslata e quindi visualizzata in modo non corretto.