Università degli Studi di Messina

Dipartimento di Ingegneria

Corso di Laurea magistrale in Ingegneria Elettronica per l'Industria

Progetto di un "Sintetizzatore FM" su FPGA

Studente:

Vincenzo Bucaria

ANNO ACCADEMICO 2023/2024

Indice

CAPITOLO

1 Introduzione

In questo elaborato si discute la progettazione su FPGA di una periferica per il soft processor AMD MicroBlaze™ capace di implementare la sintesi musicale di tipo FM. La sintesi musicale di tipo FM, popolare all'inizio degli anni 80, si basa sulla modulazione in frequenza di un generatore sinusoidale di tono con un altro generatore sinusoidale di frequenza opportunamente correlata con quella del generatore di tono. Limitandosi al caso in cui la frequenza modulate sia un multiplo intero (max 8) con la frequenza del generatore di tono, è stata progettata una periferica che implementa questo blocco funzionale dando la possibilità all'utente di generare, mediante l'interfacciamento del sistema con una tastiera MIDI, una vasta gamma di suoni. La periferica viene istruita circa i parametri che permettono di generare i suoni mediante l'ausilio di un server seriale implementato sul MicroBlaze e le routine di quest'ultimo sono state pensate per poter utilizzare (come già accennato) una tastiera midi interfacciata al sistema FPGA mediante uno script ad alto livello capace di tradurre i comandi MIDI in opportuni comandi da inviare al server seriale. I parametri vengono qui di seguito elencati.

1.0.1 Parametri utente

- il rapporto fra la frequenza modulante e la frequenza del generatore di tono;
- l'indice di modulazione;
- la frequenza del generatore di tono;

- la condizione “suono on” e “suono off”;

CAPITOLO

2 Progettazione del sintetizzatore

Con l'obiettivo finale di interfacciare il sistema ad una tastiera MIDI, è stato scelto di seguire un approccio di progettazione top-down che ben si presta nell'ambiente dell'hardware design e in particolar modo al linguaggio di descrizione dell'hardware VHDL ed ha permesso di intraprendere scelte progettuali mirate ad ottimizzare l'esperienza utente minimizzando la latenza del sistema.

2.0.1 Vista esterna del sintetizzatore

In ottemperanza alla quantità e al tipo di parametri utente e alla necessità di avere un clock e un segnale di reset in quanto si tratta di un sistema sequenziale sincronizzato, il corpo della periferica si presenta come un'entità avente 6 ingressi e 6 uscite.

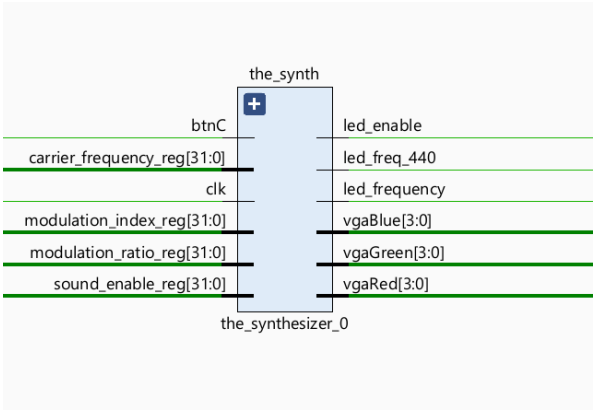

Figura 2.1: Vista esterna del sintetizzatore

I 4 vettori di (32) bit in ingresso alla periferica permettono alla stessa di ricevere le direttive dell'utente, mentre i tre vettori in uscita permettono di sfruttare i pin della VGA della Basys3 per riprodurre il suono prodotto dal sintetizzatore. Infine, le tre uscite rimanenti sono servite in fase di realizzazione come segnalazioni di debug e permettono di far accendere opportunamente dei LED.

2.0.2 Vista interna del sintetizzatore

Procedendo con un approccio top-down, possiamo osservare quali blocchi funzionali sono presenti all'interno della periferica.

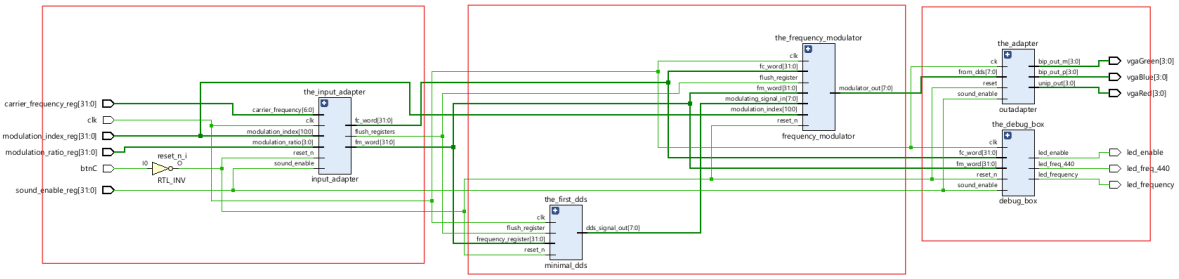

Figura 2.2: Blocchi fondamentali del sintetizzatore

I blocchi fondamentali possono essere raggruppati in tre stadi qui rappresentati dai rettangoli rossi, rispettivamente da sinistra verso destra: stadio di frontend, stadio centrale, stadio di output.

2.0.3 Stadio di frontend

Lo stadio di frontend permette di ricevere i parametri utente e manipolarli al fine di generare opportuni segnali mandati in pasto ai blocchi funzionali degli altri stadi al fine di adempire alle specifiche impostate dall'utente.

L'unica entità che compone questo stadio è l'entità "the input adapter", qui di seguito discussa

2.0.3.1 The input adapter

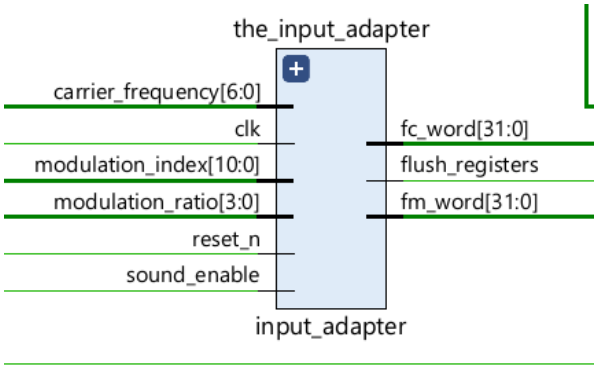

Figura 2.3: Entità Input adapter

Come si può vedere dalla figura, l'entità riceve in ingresso (oltre al segnale di reset e al segnale di clock), un segnale di tipo bit e tre array di bit. Come si può intuire dai nomi dei segnali, il segnale "sound enable" permette di attivare o disattivare la generazione del tono, il segnale "carrier frequency" codifica la frequenza del tono, il segnale "modulation ratio" codifica il rapporto tra frequenza del segnale modulante e la frequenza del tono. Nel segnale "modulation index", è invece codificato l'indice di modulazione. I valori di questi ingressi si riflettono sui valori associati ai segnali di uscita "fc word" e "fm word". L'entità si occupa anche di mandare alto il segnale "flush registers" per un colpo di clock al fine di fare il flush degli accumulatori di fase utilizzati nei DDS del sistema quando i parametri sonori vengono modificati.

2.0.3.2 Considerazioni progettuali

La scelta delle dimensioni dei segnali di ingresso parte dall'osservazione che, per ridurre la latenza del sistema, è necessario ridurre la quantità di informazione inviata e ricevuta mediante la seriale.

Per quanto riguarda il segnale "carrier frequency" erano possibili diverse scelte progettuali legate a come codificare la frequenza del generatore di tono. In prima ipotesi, si è pensato di trasmettere mediante la seriale direttamente la word capace di far generare al DDS il tono sinusoidale con frequenza corrispondente alla word specificata. Tuttavia, in relazione al fatto

che viene usato un registro di fase a 32 bit e un clock operante a 100MHz e, ponendoci nel caso peggiore, per codificare la word necessaria a generare un tono con frequenza di 4186Hz (corrispondente alla frequenza fondamentale della nota MIDI più alta utilizzabile, cioè la nota 88) sarebbero stati necessari 18 bit.

In una seconda e ultima ipotesi, si è scelto di inviare all'input adapter un segnale contenente la codifica del numero di una noda MIDI. In relazione al fatto che vengono utilizzate 88 note midi, stiamo così utilizzando solamente 7 bit. Come si può intuire, la word da mandare al modulatore e corrispondente a ciascuna nota MIDI è generata all'interno della periferica e codificata sul segnale in uscita "fc word".

Per quanto riguarda il segnale di abilitazione, si osservi che per stabilire la condizione del suono (on/off), basta ovviamente 1 bit. Il rapporto di modulazione, in quanto limitato ad 8, è ovviamente codificabile in 4 bit. Si discuterà più avanti sul perché il vettore relativo all'indice di modulazione è di lunghezza 11 bit.

2.0.3.3 Composizione dell'entità input adapter

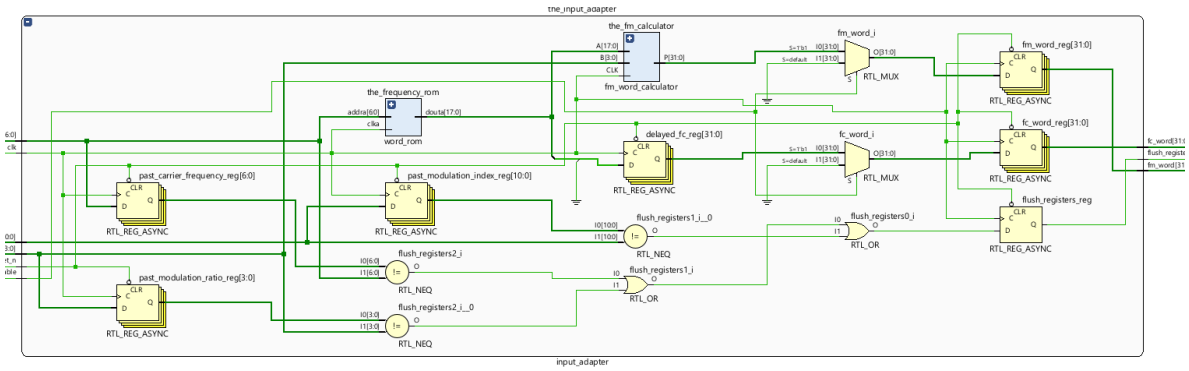

Figura 2.4: Entità Input adapter

Come già accennato, l'entità è responsabile nel ricavare le word da mandare in pasto ai DDS dello stadio centrale. Per quanto riguarda la word legata alla frequenza della portante, è stata usata una ROM che ha consentito di fare un map tra nota midi e la corrispettiva word capace di permettere a un DDS con registro di fase a 32 bit e operante su un clock di 100MHZ di generare un tono con frequenza fondamentale pari a quella della noda MIDI selezionata.

A partire da questa word, denominata ”fc word”, è possibile ricavare la word necessaria da mandare a un ulteriore DDS per generare il segnale modulante avente frequenza pari a

$$fm = fc * rapportomodulazione$$

Dunque, la word ”fm word” è ottenuta per mezzo di un moltiplicatore avente come fattori ”fc word” e il rapporto di modulazione, codificato per mezzo del segnale ”modulation ratio”.

In stato di reset, le due word sono poste a zero, identicamente a quando il bit di enable è posto a 0.

2.0.4 Stadio centrale

Presso lo stadio centrale avviene la sintesi vera e propria del suono. La prima entità a sinistra permette di generare il segnale modulante, che viene successivamente inviato alla seconda entità, la quale si occupa di effettuare la modulazione vera e propria.

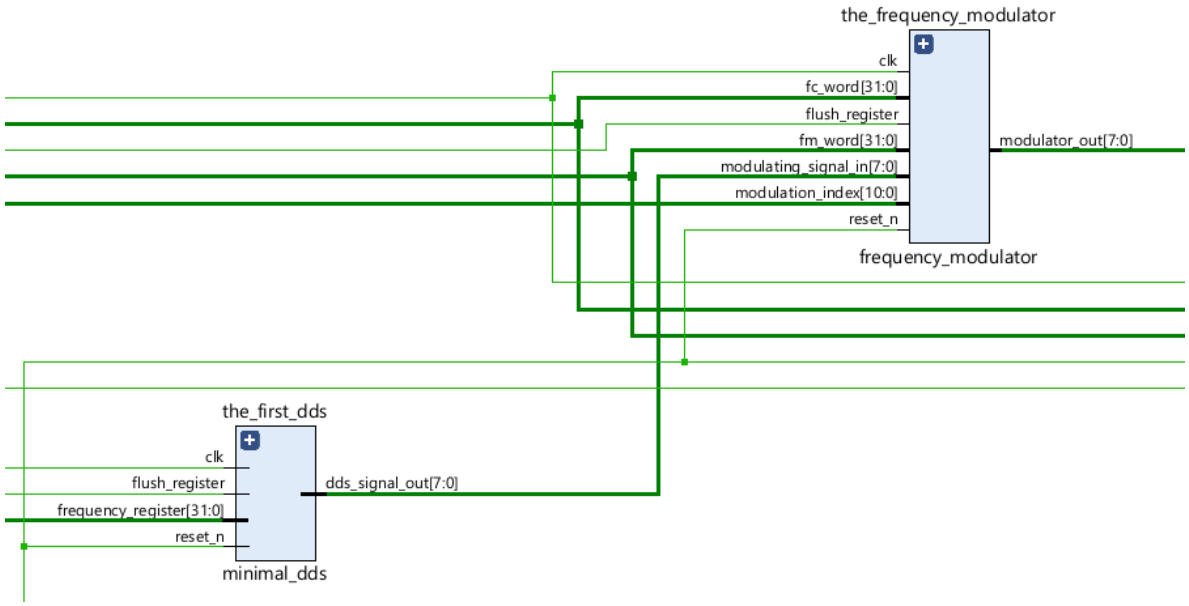

Figura 2.5: Stadio centrale

2.0.4.1 Entità minimal dds

Come già accennato, mediante questa entità è possibile generare la sinusoide modulante. Si tratta di un DDS in veste minimale, in quanto composto soltanto da un oscillatore numerico e da una ROM.

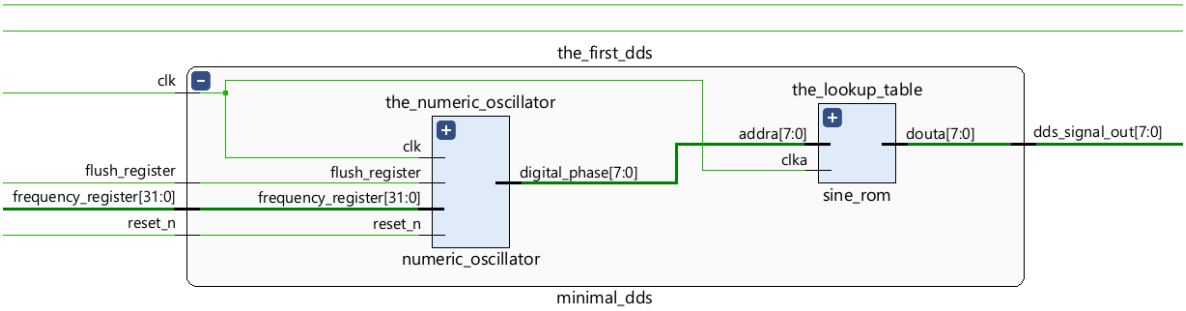

Figura 2.6: Entità minimal dds

All'interno dell'oscillatore numerico viene fatto avanzare l'accumulatore di fase a 32 bit, ad ogni colpo di clock, di un incremento pari alla parola "fm word". Il valore presente nell'accumulatore rappresenta la fase digitale della sinusoide e viene usato come indirizzo di lettura nella ROM. All'interno della ROM sono presenti le parole digitali a 8 bit che rappresentano l'ampiezza della sinusoide in complemento a 2.

La frequenza della sinusoide generata dipende direttamente dal valore della "fm word". Ad ogni colpo di clock, l'accumulatore di fase viene incrementato e quando raggiunge il valore massimo di 32 bit, va in overflow e riparte da zero. Un valore maggiore di "fm word" provoca un incremento più rapido dell'accumulatore di fase, facendo sì che vada in overflow più frequentemente. Di conseguenza, la sinusoide completa un ciclo in meno colpi di clock, risultando in una frequenza più alta. Al contrario, un valore minore di "fm word" provoca un incremento più lento dell'accumulatore di fase, allungando il tempo necessario per raggiungere l'overflow e completare un ciclo della sinusoide, generando quindi una sinusoide con frequenza più bassa.

2.0.4.2 Definizione dell'indice di modulazione

Uno degli obiettivi del progetto era la definizione del parametro indice di modulazione. Da definizione, l'indice di modulazione indica di quanto la variabile modulata cambia intorno al suo valore non modulato.

Formalmente, l'indice di modulazione è

$$\beta = \frac{\Delta f}{f_m}$$

dove

$$\Delta f = A_m k_f$$

è la deviazione in frequenza e rappresenta lo scostamento massimo tra la frequenza istantanea della portante e quella del segnale modulato.

In sintesi musicale FM, l'indice di modulazione influenza il numero di componenti spettrali nonché la loro ampiezza, determinando il timbro del suono.

Allo scopo di comprendere come permettere all'utente di variare l'indice di modulazione, sono state effettuate le seguenti osservazioni.

Esplicando la definizione di indice di modulazione si ha

$$\beta = \frac{\Delta f}{fm} = \frac{fmax - fc}{fm} = \frac{-fmin + fc}{fm}$$

Poiché è in uso un DDS, possiamo ulteriormente escplicitare

$$\frac{fmax - fc}{fm} = \frac{fcword + Am * x}{2^{32} * fm} * fclock - \frac{fcword}{2^{32} * fm} * fclock$$

Semplificando si ha

$$\beta = \frac{Am * x}{2^{32} * fm} * fclock$$

Dalla quale si può ricavare

$$x = \frac{\beta}{Am} * \frac{2^{32} * fm}{fclock}$$

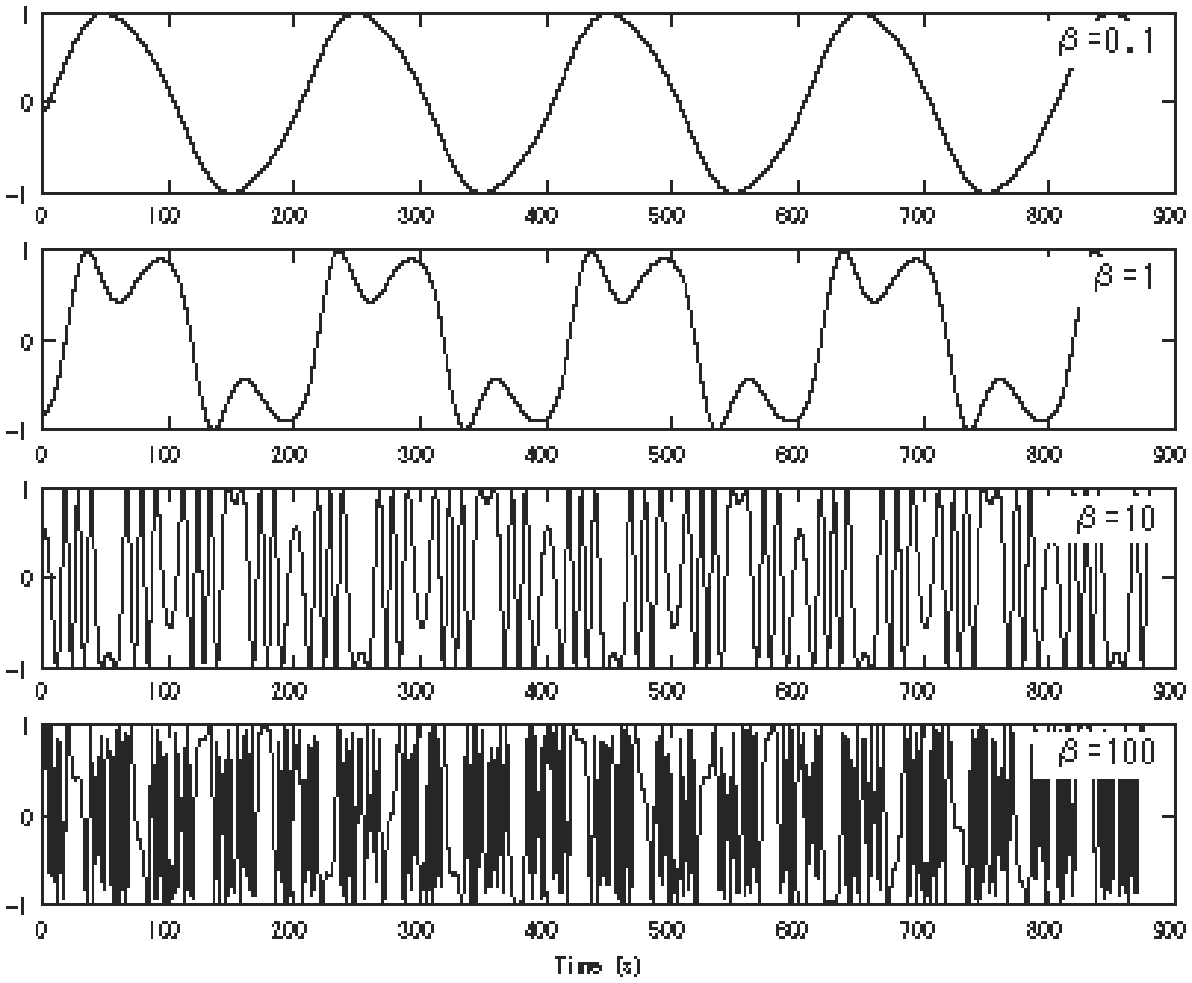

Figura 2.7: Da wikipedia: effetti dell'indice di modulazione sul timbro

Si noti che

$$\frac{2^{32} * fm}{fclock} = fmword$$

E dunque

$$x = \frac{\beta}{Am} * fmword$$

Se la frequenza istantanea del segnale in uscita dal DDS modulatore è pari a

$$f(t,x) = \frac{(fcword + m(t) * x)}{2^{32}} * 100MHz$$

dove m(t) è l'ampiezza istantanea del segnale modulante,

si può concludere che, fissati fcword e fclock, l'indice di modulazione può essere variato in funzione di x.

2.0.4.3 Entità frequency modulator

A partire dall'indice di modulazione, dalle word relative alla portante e alla modulante (quest'ultima serve per calcolare il coefficiente x prima discusso), nonché dal segnale modulante, l'entità in oggetto produce il suono sintetizzato.

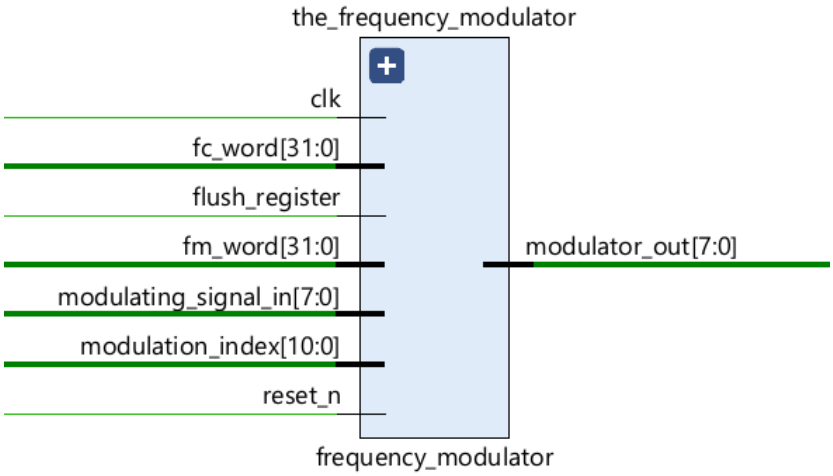

Figura 2.8: Da wikipedia: vista esterna dell'entità modulatore

Si noti come la dimensione del vettore di bit "modulation index" sia pari a 11 bit. Infatti, si è scelto di far specificare all'utente un indice di modulazione compreso tra [0.0, 100.0]. Per permettere una regolazione fine dell'indice di modulazione, è presente una cifra decimale. Per gestire la presenza di numeri razionali, si è pensato di mandare in pasto al modulatore il valore dell'indice di modulazione desiderato moltiplicato per sedici, questa operazione è compiuta dallo script di alto livello.

Per tenere conto di ciò, all'interno del modulatore, come vedremo, il coefficiente x prima discusso viene diviso per sedici eseguendo uno shift aritmetico a destra di 4 bit.

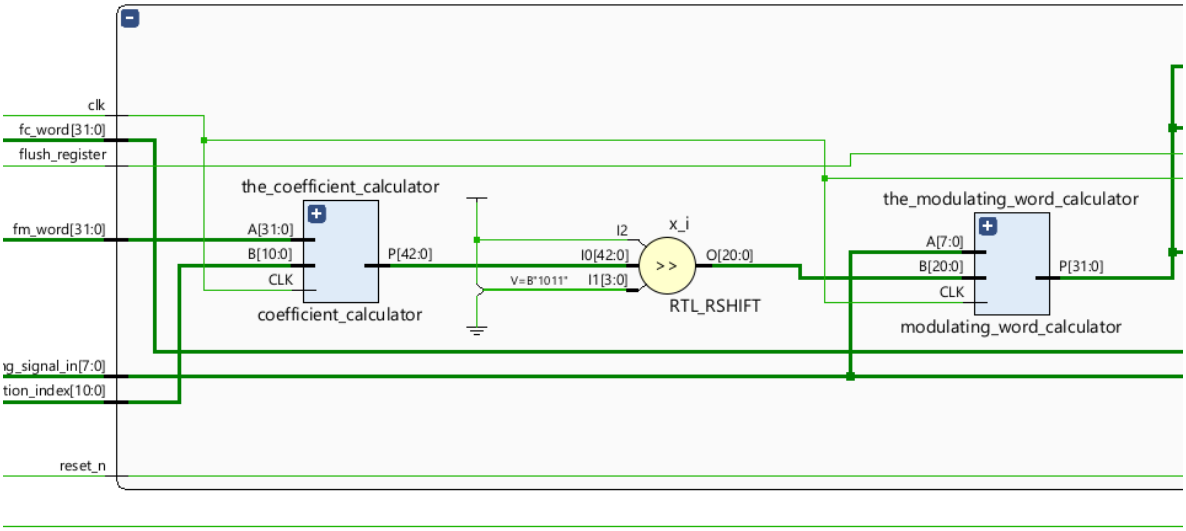

Figura 2.9: Blocchi utili al calcolo del coefficiente

All'interno dell'entità possiamo notare la presenza di un moltiplicatore e di un blocco che esegue lo shift aritmetico a destra di 11 bit.

Si noti che ciò corrisponde al calcolo del coefficiente x commettendo un'approssimazione. Infatti, considerando che l'ampiezza massima della sinusoide in modulo ed espressa in decimale è pari a 127, in uscita al moltiplicatore avremmo dovuto dividere per 127*16. Per evitare di dover istanziare un divisore, si è preferito accettare un piccolo errore e dividere per 128*16 così da sfruttare la possibilità di dividere eseguendo uno shift aritmetico a destra.

Il coefficiente moltiplica successivamente l'ampiezza istantanea della modulante così da ottenere la word che effettivamente consente di modulare il tono portante con l'indice di modulazione specificato dall'utente.

Nell'ambito di questo sistema digitale, la modulazione in frequenza avviene (così come visto prima nelle considerazioni sull'indice di modulazione) sommando aritmeticamente tale word a "fc word".

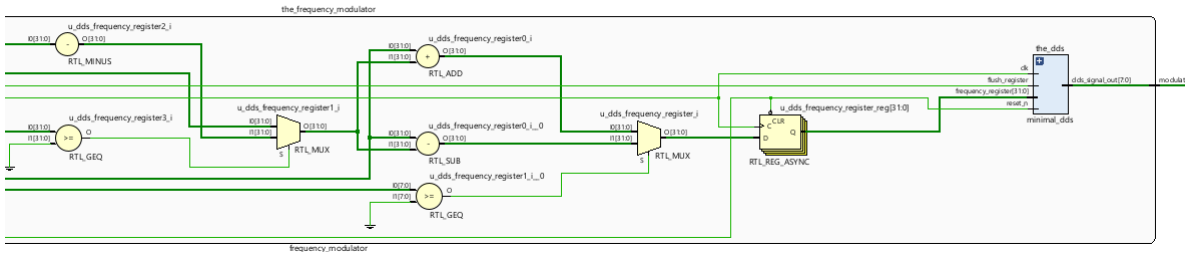

Figura 2.10: Vista interna del modulatore

La word risultante viene infine mandata in pasto a un'istanza dell'entità "minimal dds" (discussa precedentemente) cui produce i segnali digitali da mandare in pasto allo stadio di uscita per permettere la riproduzione del suono sintetizzato sfruttando il DAC della VGA.

2.0.5 Stadio di uscita

Infine, lo stadio di uscita è formato da due entità. L'entità "outadapter" è stata realizzata dal docente durante la lezione incentrata sul DDS e consente di ottenere, mediante il DAC della VGA della Basys3, 2 uscite bipolari da usare qualora si abbia un amplificatore differenziale e un'uscita unipolare. L'entità è stata lievemente modificata per porre "a zero" i segnali di uscita quando non si desidera alcun suono.

L'entità "debug box", come già detto, è servita nelle fase di realizzazione come elemento di debug. Senza entrare nei dettagli in quanto si tratta di un'entità banale, permette di accendere dei LED per segnalare i seguenti eventi:

"sound enable" allo stato logico alto; "fc word" diversa da 0; frequenza di tono impostata dall'utente pari a 440Hz.

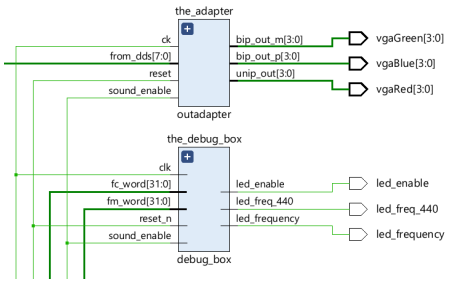

Figura 2.11: Stadio di uscita

CAPITOLO

3 Simulazione del sintetizzatore

Mediante il tool di simulazione di Vivado è stato possibile osservare il comportamento della periferica in ambiente simulato, permettendo di rilevare eventuali errori e correggerli. Poiché il simulatore non può tener conto di segnali esterni ad esso, è stato ovviamente necessario generare un testbench capace di sollecitare opportunamente la periferica.

Iniziamo ad osservare cosa viene prodotto quando per i parametri in ingresso alla periferica si specifica un tono corrispondente a LA4 (440Hz), un rapporto di modulazione pari a 2 e un indice di modulazione pari a 0.

In figura 3.1 è presente uno screenshot della simulazione. Visivamente si può notare come il segnale in uscita al modulatore (”modulated signal”) abbia una frequenza dimezzata rispetto alla frequenza del segnale modulante (”modulating signal”). Ed è in effetti questo il comportamento atteso in quanto abbiamo, come parametro, un rapporto di modulazione pari a 2. Si noti, inoltre, che il segnale in uscita al modulatore non appare affatto modulato, ma anche questo è il risultato atteso in quanto abbiamo impostato un indice di modulazione pari a 0.

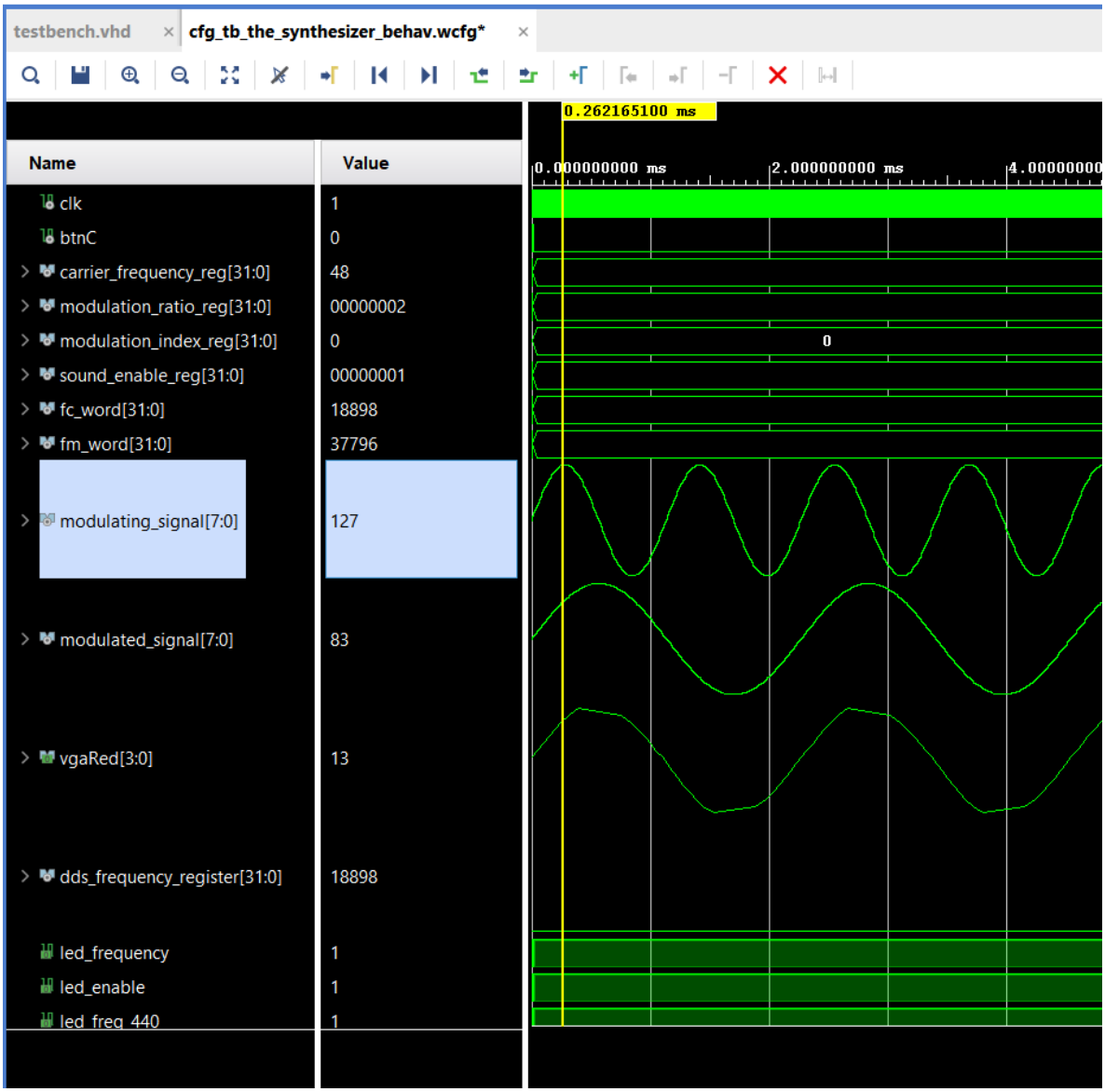

Figura 3.1: Simulazione con indice di modulazione pari a 0

Per osservare gli effetti della modulazione, spostiamo la timeline di 5ms, ovvero presso il frangente temporale nel quale il testbench varia l'indice di modulazione portandolo a 0.125.

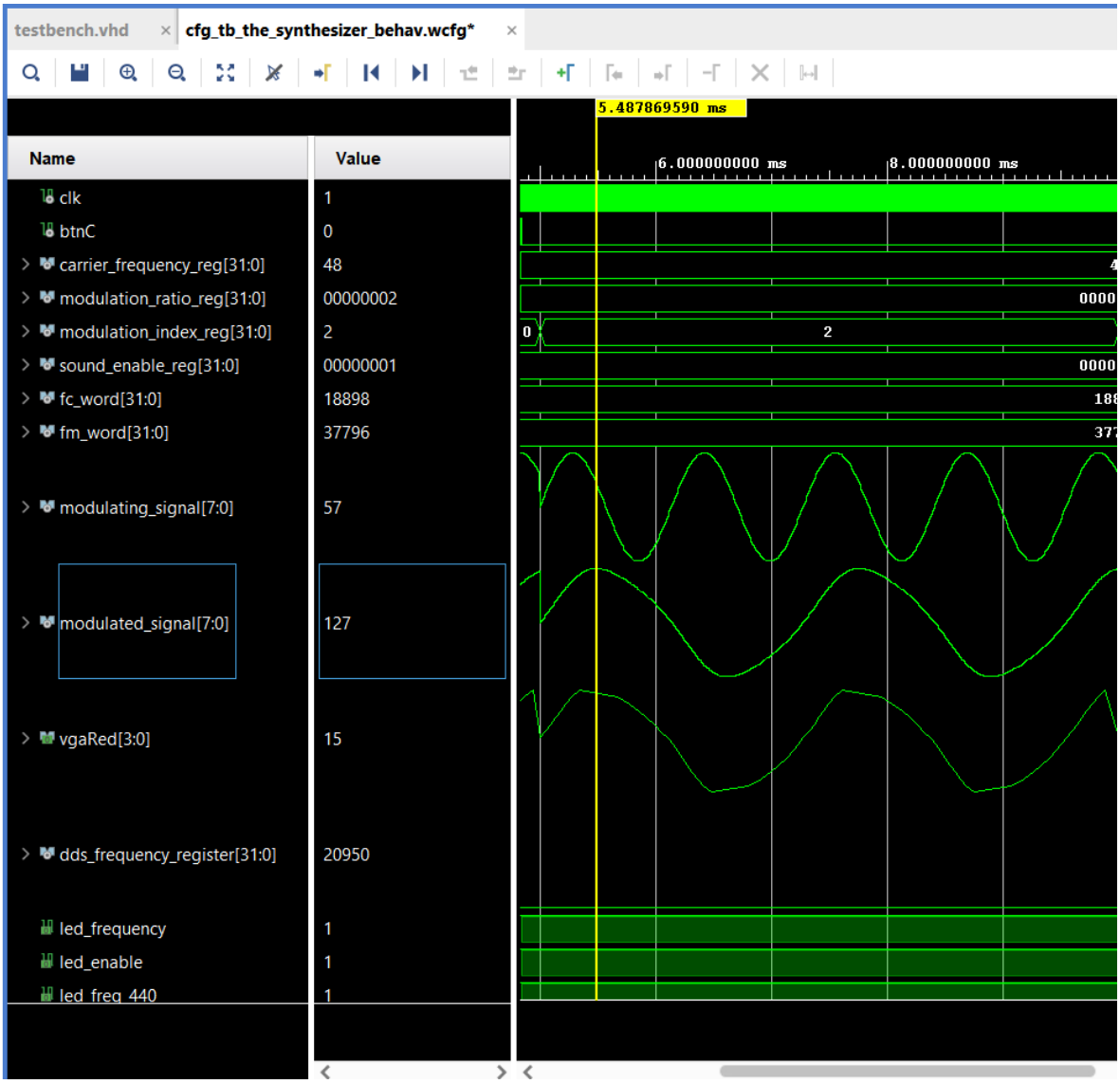

Figura 3.2: Simulazione con indice di modulazione pari a 0,125

Come si può notare, il segnale modulato non è più una sinusoide pura per effetto della (debole) modulazione. In riferimento alla figura 2.7, si può osservare come il segnale modulato prodotto sia identico a quello rappresentato da Wikipedia (ottenuto, ovviamente, con i medesimi parametri).

Spostando la timeline a 10 ms, possiamo vedere chiaramente, in figura 3.3, gli effetti della modulazione quando l'indice di modulazione è variato a 1.

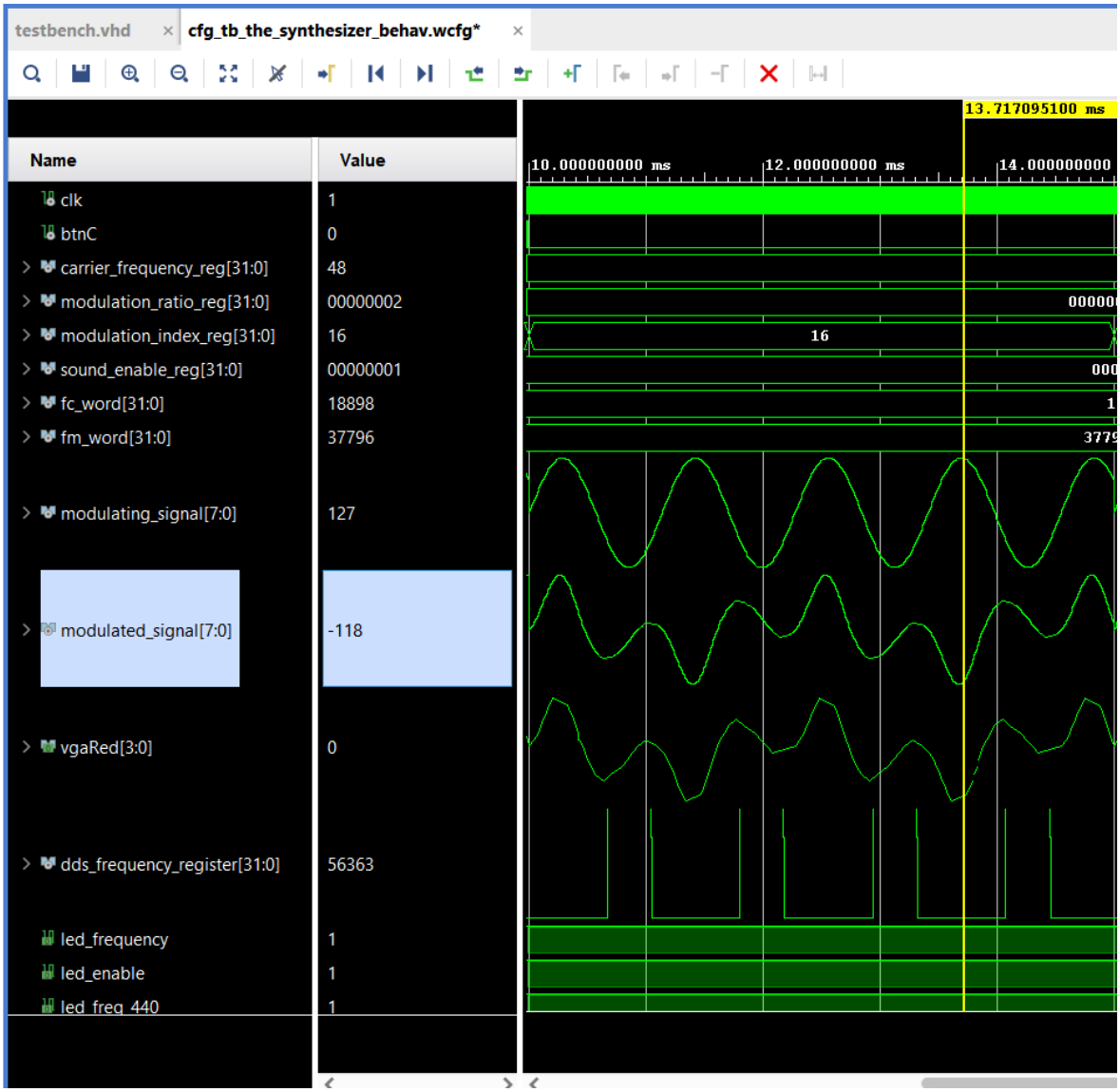

Figura 3.3: Simulazione con indice di modulazione pari a 1

La modulazione adesso appare evidente ed è in accordo a quanto mostrato in figura 2.7. Servendoci della simulazione, possiamo calcolare, a partire dal segnale "dds frequency register", la deviazione massima di frequenza che si verifica in fase di simulazione ed ottenere dunque l'indice di modulazione effettivo.

Si può osservare che quando l'ampiezza del segnale modulante è massima (+127), la word contenuta nel segnale "dds frequency register" può essere rappresentata in decimale come

56068 ed è la word in corrispondenza della quale si ha la frequenza di picco fmax.

Possiamo dunque calcolare la frequenza massima come

$$fmax = \frac{56068 * 100 * 10^6 Hz}{2^{32}} = 1305Hz$$

Dalla quale si può ricavare l'indice di modulazione

$$\beta = \frac{fmax - fc}{fm} = \frac{1305Hz - 440Hz}{880Hz} = 0.9829 \simeq 1$$

Osserviamo invece cosa accade quando l'ampiezza del segnale modulante è minima (-127). In questo caso la somma tra la word modulante e la word della portante produce un risultato che rappresentato in decimale è 4294948729.

Analiticamente in uscita al DDS si otterrebbe

$$fteorica = \frac{4294948729 * 100 * 10^6 Hz}{2^{32}} = 99999567,7Hz$$

Poiché il teorema del campionamento non è rispettato, in realtà il DDS produce un segnale con frequenza istantanea pari a

$$f = fclock - fteorica = 100MHz - 99999567,7Hz = 432,29Hz.$$

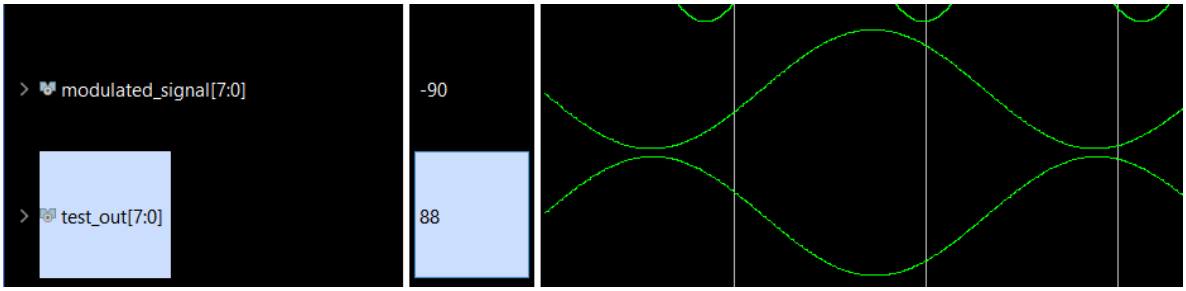

Figura 3.4

e con fase opposta a quella che si otterrebbe generando una sinusoide con frequenza di 432,29Hz rispettando il teorema del campionamento.

Si osservi la figura 3.4, il segnale ”modulated signal” è prodotto mandando in pasto ad un DDS la word (rappresentata in decimale) 4294948729. Il segnale ”test out” è prodotto mandando in pasto ad un DDS la word (rappresentata in decimale)

$$testword = \frac{2^{32} * 432,29}{100MHz} = 18567$$

Possiammo modellizzare questo ribaltamento di fase dicendo che

$$fmin = -f = -432,29Hz.$$

Otteniamo dunque un indice di modulazione

$$\beta = \frac{-fmin + fc}{fm} = \frac{-(-432,29Hz) + (440Hz)}{880Hz} = 0.991 \simeq 1$$

Spostandoci a t=15ms, l’indice di modulazione è variato a 10 e infatti si può apprezzare come il segnale generato sia ancora più complesso.

Per osservare come anche il rapporto di modulazione possa incidere sul segnale modulato, si può spostare la timeline al tempo t=20ms, ovvero presso l’istante temporale in cui il testbench varia il rapporto di modulazione a 6 e l’indice di modulazione a 1.

In figura 3.5 è possibile osservare il segnale modulato prodotto in fase di simulazione.

Infine, se ci si sposta a t=25ms, ovvero l’istante temporale in cui il testbench porta allo stato logico basso il parametro ”sound enable”, si può notare come in uscita al DAC ci sia il valore nullo, così come atteso.

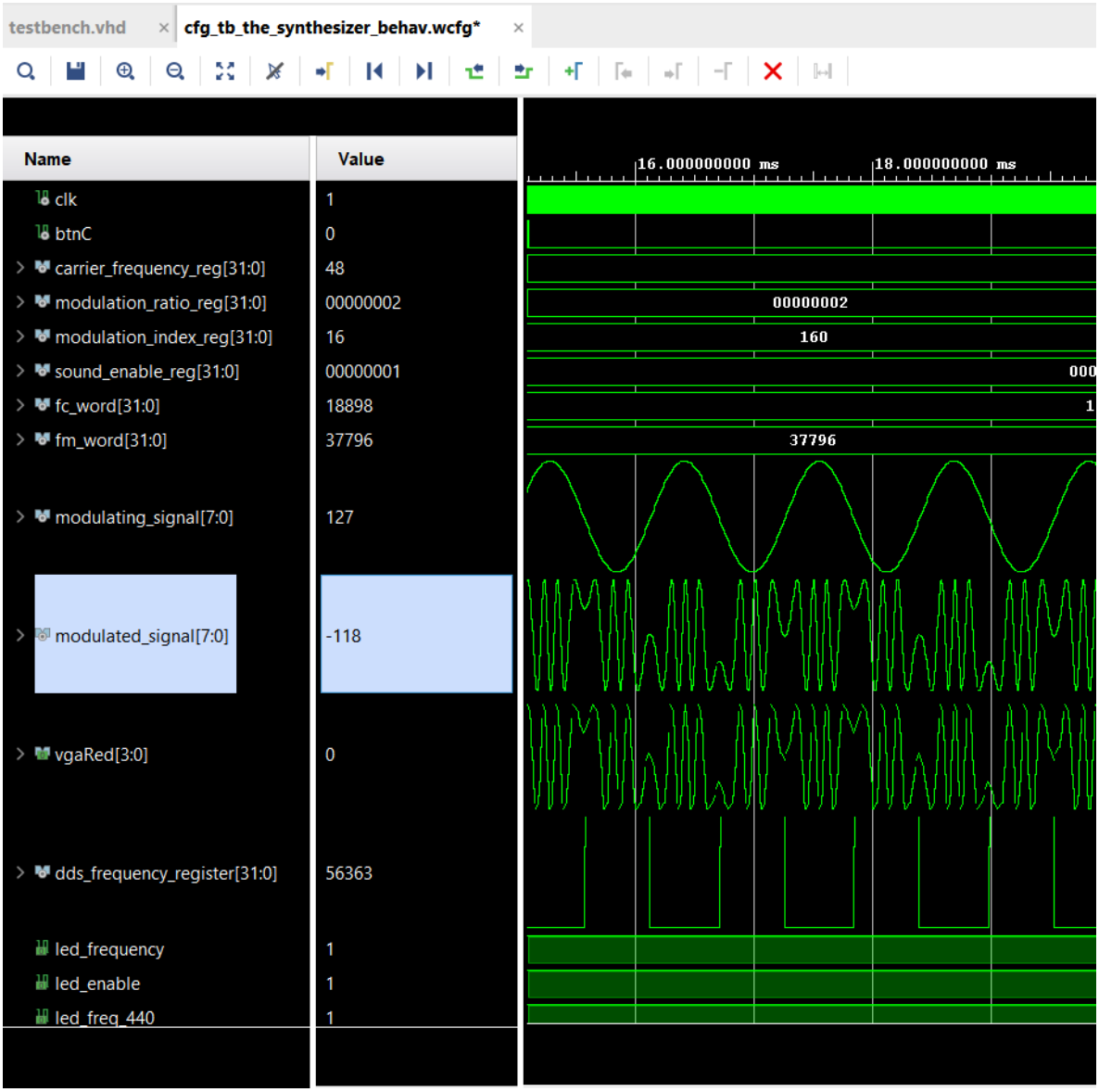

Figura 3.5: Simulazione con indice di modulazione pari a 1

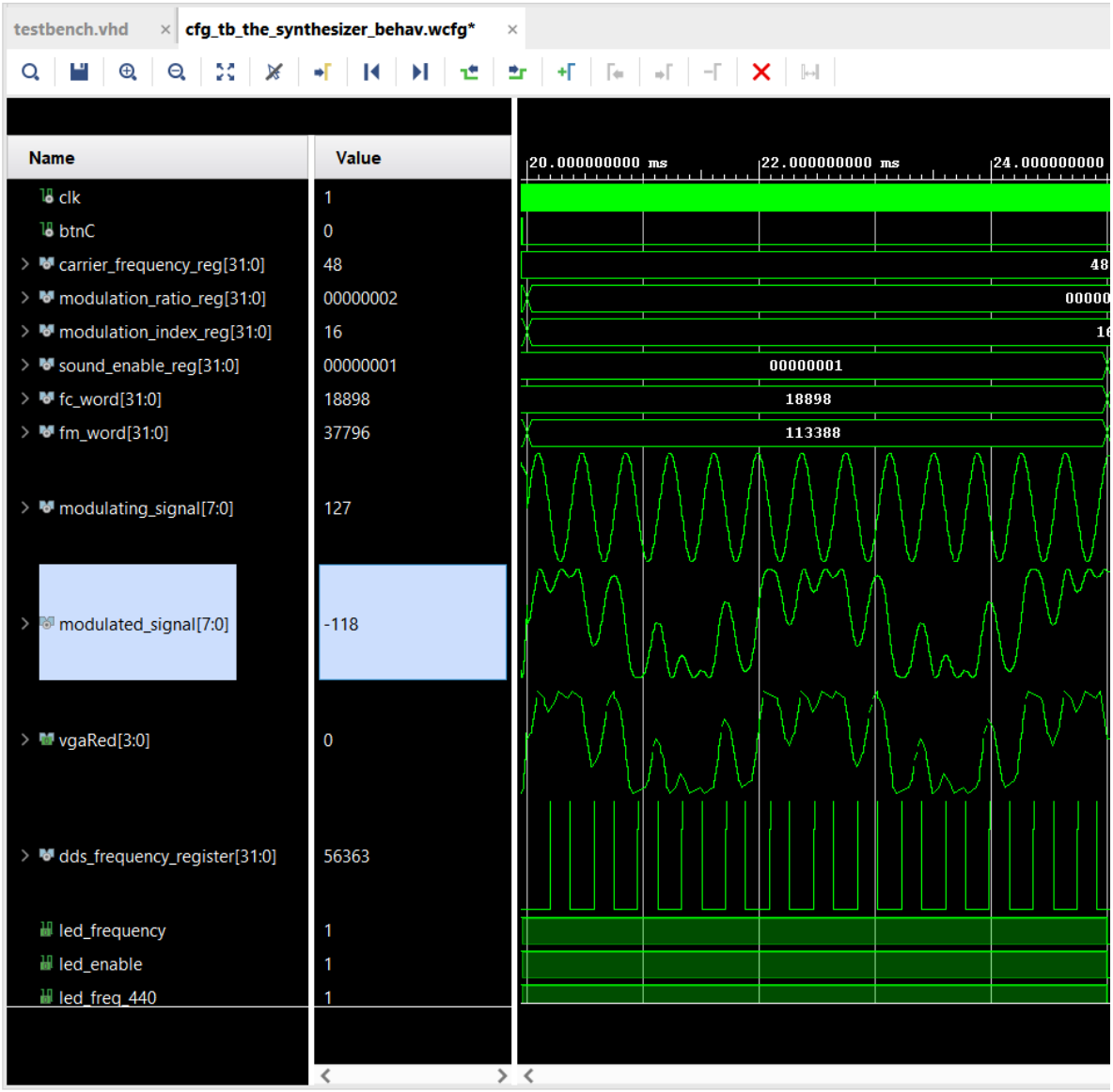

Figura 3.6: Simulazione con indice di modulazione pari a 1 e rapporto di modulazione pari a 6

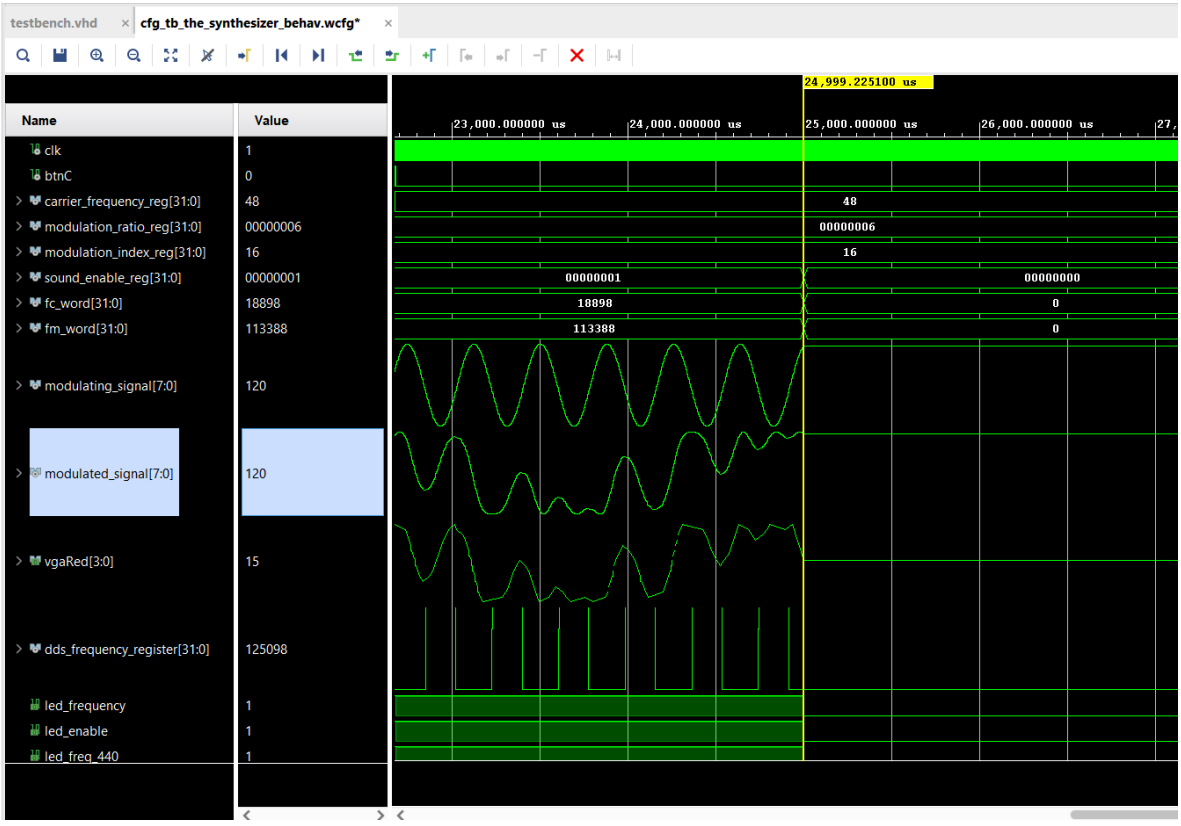

Figura 3.7: Simulazione parametro ”sound enable” allo stato logico basso

CAPITOLO

4 Sintetizzatore come periferica per AMD MicroBlaze™

Appurato il corretto funzionamento del corpo della periferica mediante le simulazioni, si è proceduto ad integrare il sintetizzatore come parte di un System on Chip (SoC) basato sul soft processor AMD MicroBlaze™.

4.0.1 Sintetizzatore come periferica su bus AXI

Per comodità è stato innanzitutto creato un IP package del corpo della periferica (disponibile nella directory ”ip repo”).

Successivamente, poiché Il SoC si basa su un'architettura che impiega il bus AXI per l'interfacciamento dei vari moduli periferici e core di elaborazione, è stata seguita la procedura guidata di Vivado che consente di creare un'IP del tipo ”AXI4 Peripheral” la quale in prima istanza conterrà la logica di interfacciamento con il bus AXI.

E' stata innanzitutto modificata l'interfaccia della periferica appena creata in quanto non solo dovrà contenere tutte le porte di interfacciamento con il bus AXI, ma dovrà contenere anche le porte di uscita del sintetizzatore.

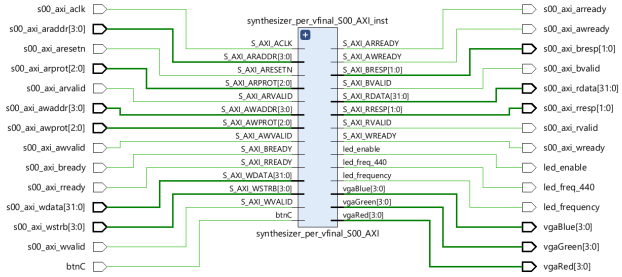

Figura 4.1: Simulazione con indice di modulazione pari a 0

Successivamente è stata istanziata l'IP del sintetizzatore all'interno della periferica AXI4. Ovviamente i quattri ingressi relativi ai parametri dell'utente sono stati collegati ai registri della periferica, il segnale di clock è stato anch'esso prelevato dall'interfaccia AXI. Le porte qui non nominate sono state collegate a cascata alle porte visibili all'esterno della periferica AXI.

4.0.2 Integrazione della periferica al SoC

Per comodità è stato riutilizzato il file di progetto fornito dal docente (UB33) e mediante il tool grafico "Block Design" di Vivado è stato ripulito delle periferiche non necessarie a questo progetto e sono state variate le proprietà della periferica "AXI UART lite", variando il baudrate da 9600baud/s a 115200baud/s.

Successivamente è stato usato il tool "IP integrator" di Vivado per istanziare la periferica AXI appena creata e opportunamente modificata all'interno del SoC. Una volta permesso a Vivado di eseguire automaticamente le interconnessioni tra gli elementi del SoC e prestando attenzione nel definire le porte non inerenti al bus AXI della nostra periferica come "external", il sistema nel suo complesso si presenta come in figura 4.2.

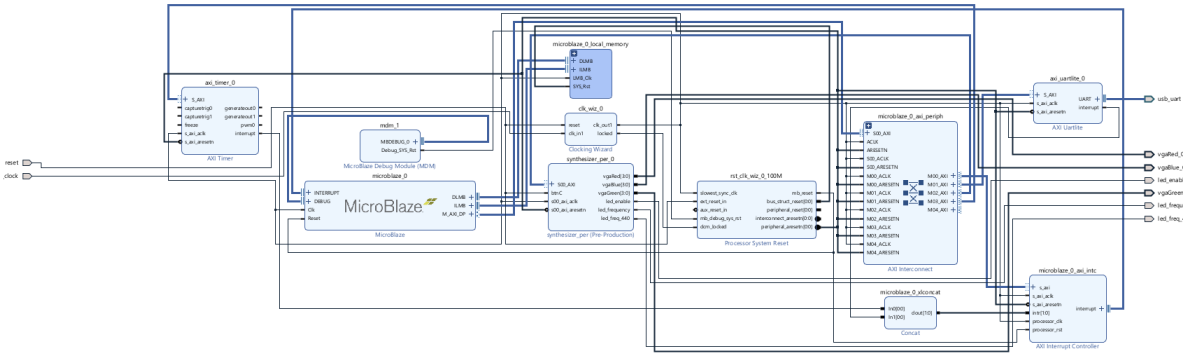

Figura 4.2: SoC

A questo punto è stato sufficiente creare un wrap HDL del block design e procedere con la generazione del bitstream. Ad operazione terminata, è possibile esportare l'hardware (includendo il bitstream) per poi poter procedere alla modifica del firmware del SoC.

4.0.3 Firmware

Sono state effettuate alcune modifiche al server UART sviluppato dal docente. Innanzitutto è stato aggiornato il baud rate della UART, successivamente sono state aggiunte delle routine per permettere allo script di alto livello di impostare i parametri utente. Nella command list sono infatti adesso presenti i seguenti comandi

```
struct command cmdlist[]={
      {"NOP",0,ex_nop}, //no operation
      {"WRP",1, ex_wrp}, //write a number of dots (".....") specified by
         the arg
        {"RTO",1,ex_ratio},
        {"IDX",1,ex_index},
        {"NOT",2, ex_note}
};
```

I comandi "RTO" e "IDX" permettono rispettivamente di impostare il rapporto di modulazione e l'indice di modulazione. Il comando "NOT", avente due argomenti, permette di

impostare la condizione suono off/suono on e la frequenza della portante.

Poiché il parametro con codifica più lunga necessita di 3 caratteri esadecimali, è stato necessario modificare il parser come quanto segue.

```
int get_arg_val(void)
{
   int i;
   char c;
   int val=0;
   if (strlen(&cmd_string[cmd_scan])<3)
   {
      return -1;//stringa troppo corta per contenere un argomento
          (argoomento 2 cifre esadecimali conlettere maiuscole)
   }
   else
   {
      for (i=0; i<3; i++)
      {
         c=cmd_string[cmd_scan];
         if (((c>='A') && (c<='F'))||((c>='0') && (c<='9')))
         {
            if ((c>='A') && (c<='F'))
            {
               val=val*16+(c-'A')+10;
            }
            else
            {
               val=val*16+c-'0';
            }
         }
         else
         {
```

```
            return -1;//error in argument
          }

          cmd_scan++;
        }

        return val;
      }
  }
```

Tutti i parametri inviati dallo script di alto livello sono codificati, infatti, in tre caratteri esadecimali.

Infine, si riportano le funzioni che si occupano di scrivere sui registri della periferica.

```
    void ex_ratio(void) // write ler red (two args)
{
   volatile u32 * pledr=(u32 *)XPAR_SYNTH_PER_0_FREQ;
   *(pledr+1)=in_arg_val[0];
   sprintf(rsp_string,"[OK]");
    return;
}
void ex_index(void) // write ler red (two args)
{
   volatile u32 * pledr=(u32 *)XPAR_SYNTH_PER_0_FREQ;
   *(pledr+2)=in_arg_val[0];
   sprintf(rsp_string,"[OK]");
    return;
}


void ex_note(void) // write ler red (two args)
```

```
{
   volatile u32 * pledr=(u32 *)XPAR_SYNTH_PER_0_FREQ;
   *pledr=in_arg_val[0];
    *(pledr+3)=in_arg_val[1];
    sprintf(rsp_string,"[OK]");
    return;
}
```

4.0.4 Script ad alto livello

Per utilizzare il sintetizzatore appena creato mediante una tastiera MIDI, è stato creato un semplice script Python basato sulla libreria ”mido”. Mediante l'impiego di due thread, lo script scansiona continuamente l'input da terminale e i messaggi ricevuti dalla tastiera midi.

Mediante i comandi da terminale è possibile impostare il rapporto di modulazione e l'indice di modulazione. I parametri così ottenuti vengono elaborati (per quanto detto nella discussione del progetto, l'indice di modulazione deve essere moltiplicato per 16) e pacchettizzati, infine inviati mediante la seriale al sistema realizzato.

Alla ricezione di un messaggio MIDI da parte della tastiera, lo script estrae dal messaggio i parametri di nostro interesse, ovvero la nota fondamentale e la condizione suono on/off. Raccolti i parametri, questi vengono formattati e pacchettizzati, dunque inviati al sistema mediante la seriale.

CAPITOLO

5 Test del sistema reale

In conclusione, si è voluto osservare il comportamento della periferica in ambiente reale. Per fare ciò, è stato utilizzato lo script precedentemente citato, una tastiera MIDI e un mixer per poter acquisire il segnale in uscita alla Basys3.

L'acquisizione del segnale da parte del mixer ha permesso di usufruire del programma "SoundScope" (SoundScope) per visualizzare su PC le forme d'onda prodotte al variare dell'indice di modulazione e del rapporto di modulazione.

A titolo di esempio, in figura 5.1 è riportata la forma d'onda visualizzata mediante il programma utilizzando un indice di modulazione pari a 1 e un rapporto di modulazione pari a 2.

Tralasciando agli effetti dovuti alla bassa risoluzione del DAC, è possibile osservare come il risultato in ambito reale coincide con quello simulato in figura 3.3.

Infine, è stato realizzato un piccolo video disponibile all'indirizzo (Video dimostrativo)

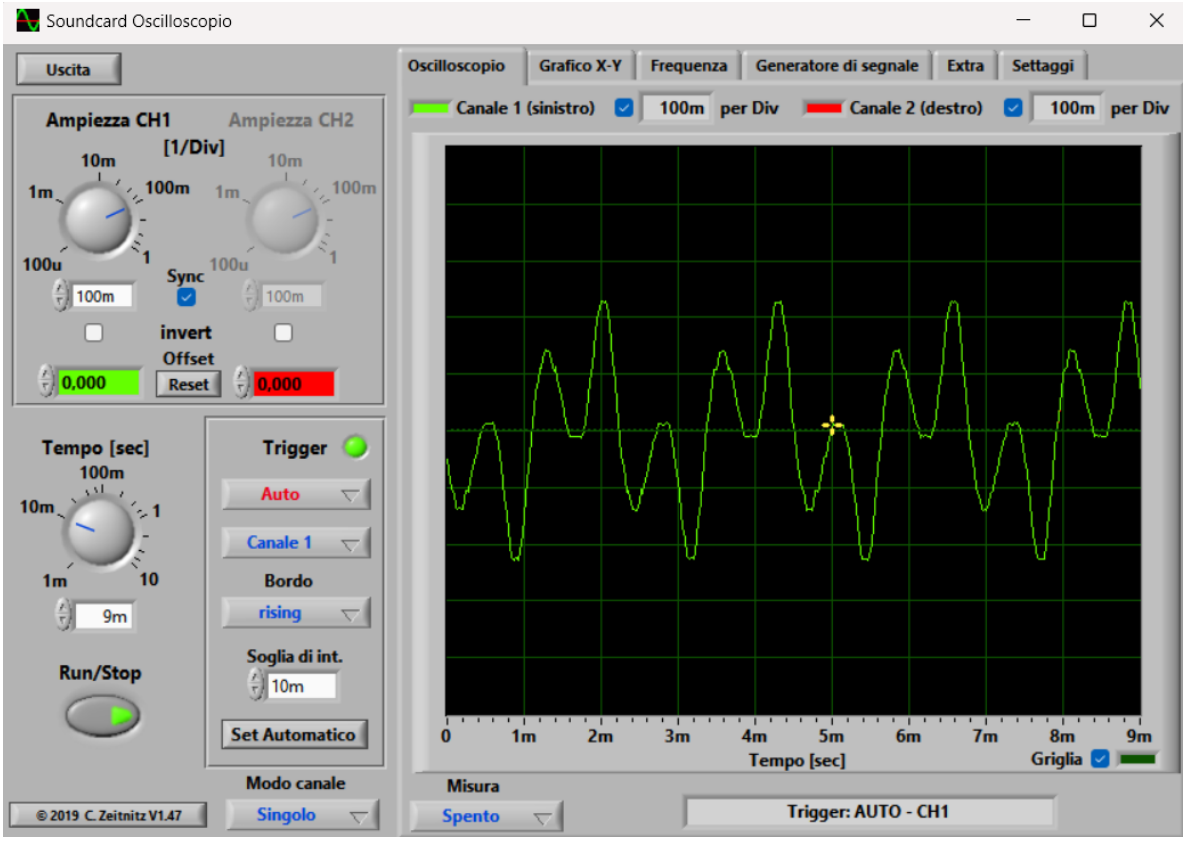

Figura 5.1: Forma d'onda visualizzata con SoundScope