SpeakING

Brandon Wu[1], Yi-Chen Chang[2], and Shih-Chieh Lin[3]

Department of Electrical Engineering, National Taiwan University
No.1, Sec. 4, Roosevelt Rd., Da'an Dist., Taipei City 106, Taiwan (R.O.C.)

[1]b99901103@ntu.edu.tw
[2]b99901023@ntu.edu.tw
[3]b99901100@ntu.edu.tw

Abstract—**SpeakING is a good helper to those having troubles in singing — if you can Speak, you can SING. It is because SpeakING can transform normal speech into a melody you create. Moreover, SpeakING can be more entertaining when combined with specifically designed sound effects. This article will introduce the theorem and the circuit realization of SpeakING on FPGA board.**

***Keywords*—vocoder; channel vocoder; voice changer; sing; pitch**

I. Introduction

In the circuit of SpeakING, there are 4 parts, channel vocoder, recorder, effects units and display. Channel vocoder can modulate the pitch of the sound; recorder can record and play back and forward the sounds; in effects units, there are some kinds of sound effects; display can show the instantaneous state when using. The first part is the most major and the most related to efficiency when realized on FPGA board, and thus we are going to put more effort to discuss the circuit of it and theorem used in it. Then, we will introduce the other parts.

The circuit diagram of the typical channel vocoder is showed in Fig. 1.

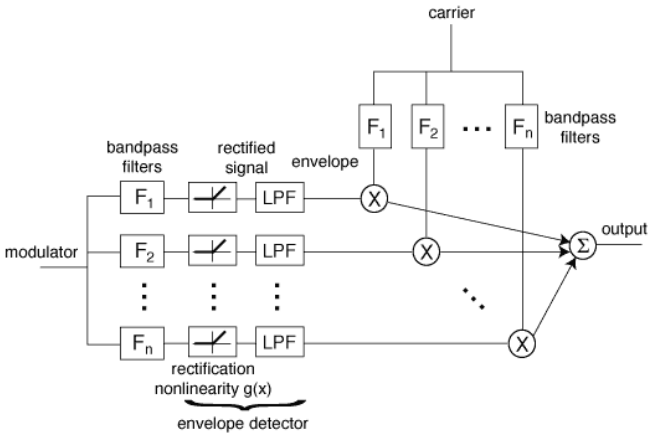


Fig. 1 Circuit diagram of the typical channel vocoder using filters[3]

In Fig. 1, a modulator, i.e., a vocal signal, will be sent to several bandpass filters at first. It passes through different envelope detectors, each of which consists of a lowpass filter and a rectifier, in order to use the envelope of it. On the other hand, a carrier, i.e., a signal of certain frequency, will also pass through the same bandpass filters as those which a modulator pass through, and thus can produce different frequency parts of a carrier. We multiply the parts of a carrier and the envelopes of a modulator individually and sequentially. Ultimately, all the results are added as the output[1]. However, this circuit ends up with a large circuit area because each filter takes up much area to realize, let alone numerous filters.

Therefore, we consider another method of realizing the circuit of channel vocoder. Both the modulator and the carrier on the time domain are transformed by the use of Fourier transform, consequently being on the frequency domain. The circuit not only saves resources but also has a better performance on efficiency.

In addition, SpeakING has basis function as a recorder, recording, playing back and forward of many states, removing data, pausing. Effects units of SpeakING include sound effects such as echo, ghost sound (sounds like an inharmonic semitone), beat cut, etc. Last but not the least, the display of SpeakING shows a piano keyboard according with the physical keyboard you use, a spectrogram of the vocal signal, and our logo "SpeakING".

II. Theorem We Apply in Channel Vocoder

There are several differences between the method of using filters and that of using Fourier transform. The former can let the signal pass through filters without interruption, while the latter needs to cut the signal into several pieces for the sake of FFT (Fast Fourier Transform).

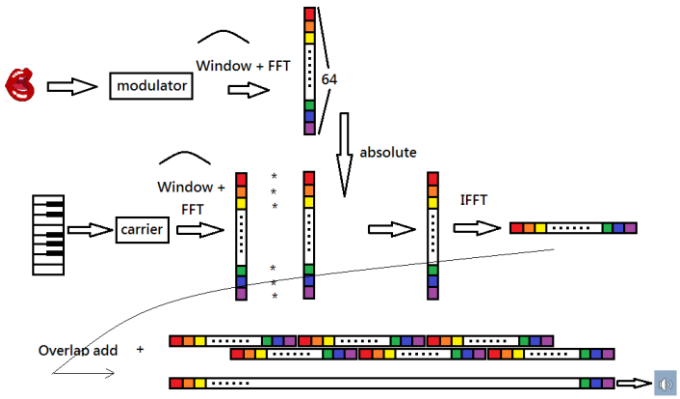


Fig. 2 System architecture of channel vocoder using FFT

In Fig. 2, our architecture takes 64 samples for a segment to compute FFT (N = 64). In order to eliminate the boundary discontinuities, we overlap the output voice for 50% and also use hanning windows.

At each segment on the frequency domain, there are 64 modulators and carriers individually. We multiply both of them one-to-one separately and this is the same situation in Fig. 1. However, we couldn't multiply the corresponding complex numbers in the frequency domain because the modulators had been envelope detected. Only using the absolute values of modulators (that is X(k), k = 0, 1, 2, … , 63) can be multiplied with the carriers. That is to say, we only focus on the magnitude rather than the phase information. That is to say, we modify the weights of the carrier signals (pitch signal) according to the modulator signals (vocal signal). After computing, the signals are translated back to the time domain by IFFT (Inverse Fast Fourier Transform) and become the output signals which are composed with the pitch we want.

III. Circuit Realization of Speaking

A. The core circuit - channel vocoder

The circuit architecture is essentially based on Fig.2 (screen display output is not shown). Each part will be introduced. Channel vocoder schematic is provided in the appendix for reference.

Input devices for the system are separated into voice input and pitch input, namely, a microphone and the pitch control keyboard. The sampling rate for microphone input is about 8kHz, and every 64 continuous sample points are stored in the buffer register, to be delivered to the FFT operation. For the keyboard input, we designed a simple circuit using 25 key-type switch and batteries for additional power supply. Whenever a certain key combination is pressed, a module (called "carrier generator") will calculate the resulting carrier waveform. This carrier signal will be sent in parallel with the voice signal to FFT modules, in a buffered way too. Electrical elements used are shown in the following diagram, Fig. 3:

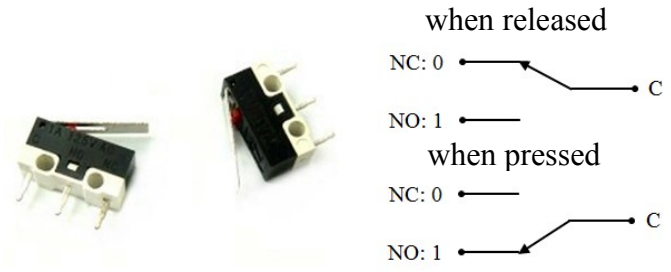


Fig.3 Switching element

When the element is pressed, C and NO are connected, C and NC are disconnected. The opposite is the case when the element is released. So we can set NO at a high potential (with two 1.5V batteries, the actual potential is between 3.1V to 3.2V) and NC ground, and have C connected to the HSMC's PIN, then any combination of key press can be related to a set of signal.

The FFT computation (also, IFFT operation) in the circuit is implemented with the built-in Mega Core Function in Quartus. The computation time is less than 0.1 ms (working at 3.125MHz), so the majority of the overall delay stems from audio buffering, which is about tens of milliseconds. That is the essence of our circuit; one can achieve real-time speech-sing conversion without any post-production.

For the remaining operations including absolute value, window function (hanning window) and the overlap on the final stage, only simple algebraic operations are used.

In addition to audio output, we also provide graphical output via VGA port on FPGA board. By connecting it to the screen, users are able to see visualized key-tone effects and audio equalizer.

B. Recording, playback , clear , fast-forward, rewind, sound effects

1) Recording: Analogue voice signal is sampled by WM8731 at about 8kHz sampling rate. Each sample is a 16-bit digital signal and is stored in the SRAM in order.

2) Play: Data in the SRAM are read as samples in the order in which the samples are stored. The samples are then converted into an analogue signal by WM8731, and the signal is the output of the speaker.

3) *Clear:* All data in the SRAM are set to 0.

4) *Fast-Forward, rewind:* Data in SRAM are read in a skipping way, some of the data will be jumped either forward or backward, depends on the multiple. It is essentially a zero-order interpolation method. Again, the resulting samples will be converted to voltage signal via WM8731, then played through the speaker.

5) *Sound effects:*

- Beatcut: Throughout all samples, we set some of them to zeros periodically. The effect is like "cutting" the voice to pieces.
- MachineCut: It is equivalent to Beatcut, but at a much higher rate of inserting zero samples. This "on and off"(equivalent to multiplying a square wave) rate has reached the frequency range of human voice, so the effect is particularly evident for human voice; it will make voice sound more hoarse, or like a robot.
- Echo: Samples delivered to two channels are slightly different in time, creating a delay, echo effect.
- GhostSound/NormalSound: Ghost sound is created by raising the pitch of one of the channels slightly, and playing two channels simultaneously. It will sound dissonant and unnatural. To achieve this, we modified the address WM8731 should read linearly and periodically. Changing the "phase" linearly with time means changing the frequency (but also playback speed), and in order to make sounds from two channel consistent in time, periodic manipulation is needed.

IV. PHYSICAL BODY AND PERFORMANCE ANALYSIS

Below, Fig.4 shows the picture of the physical body of SpeakING without the microphone appearing.

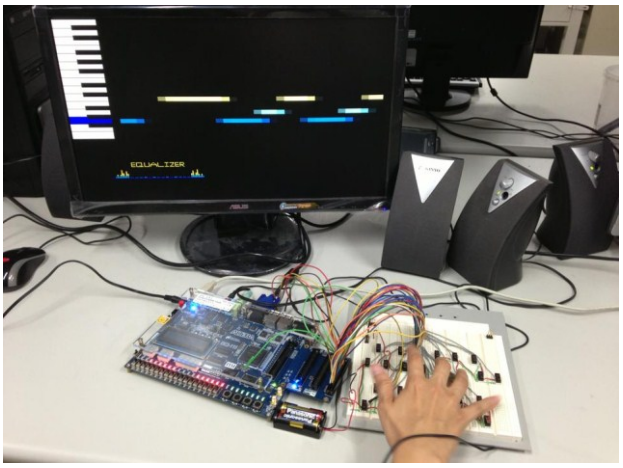


Fig. 4 Physical body of SpeakING

On the performance of efficiency, channel vocoder has the most influence. To optimize the circuit, we used MATLAB to make some simulations of channel vocoder. Firstly, not using hanning windows spares little circuit area and results in obviously low tone quality. Secondly, not overlapping leads to the incontinuous output but saves half time circuit area with the 50% overlap; overlapping 75% makes the output smoother, improves the tone quality, but brings about twice circuit area with the 50% overlap. Therefore, we choose the 50% overlap in our channel vocoder.

In addition, if we adopt the circuit architecture of the typical channel vocoder, there will be more than one hundred filters in our channel vocoder, which leads to unacceptably-huge circuit area. On the other hand, the filters provided by Mega Core Function are FIR filters instead of IIR filters which we used in MATLAB simulations, which causes the different total numbers of used elements. And different filters having their own delays will make the output addition rather difficult.

V. CONCLUSIONS

With SpeakING, people can easily not only sing but also create a lovely song. Moreover, our method of realizing SpeakING spares much circuit area, which also means spares much money to generate.

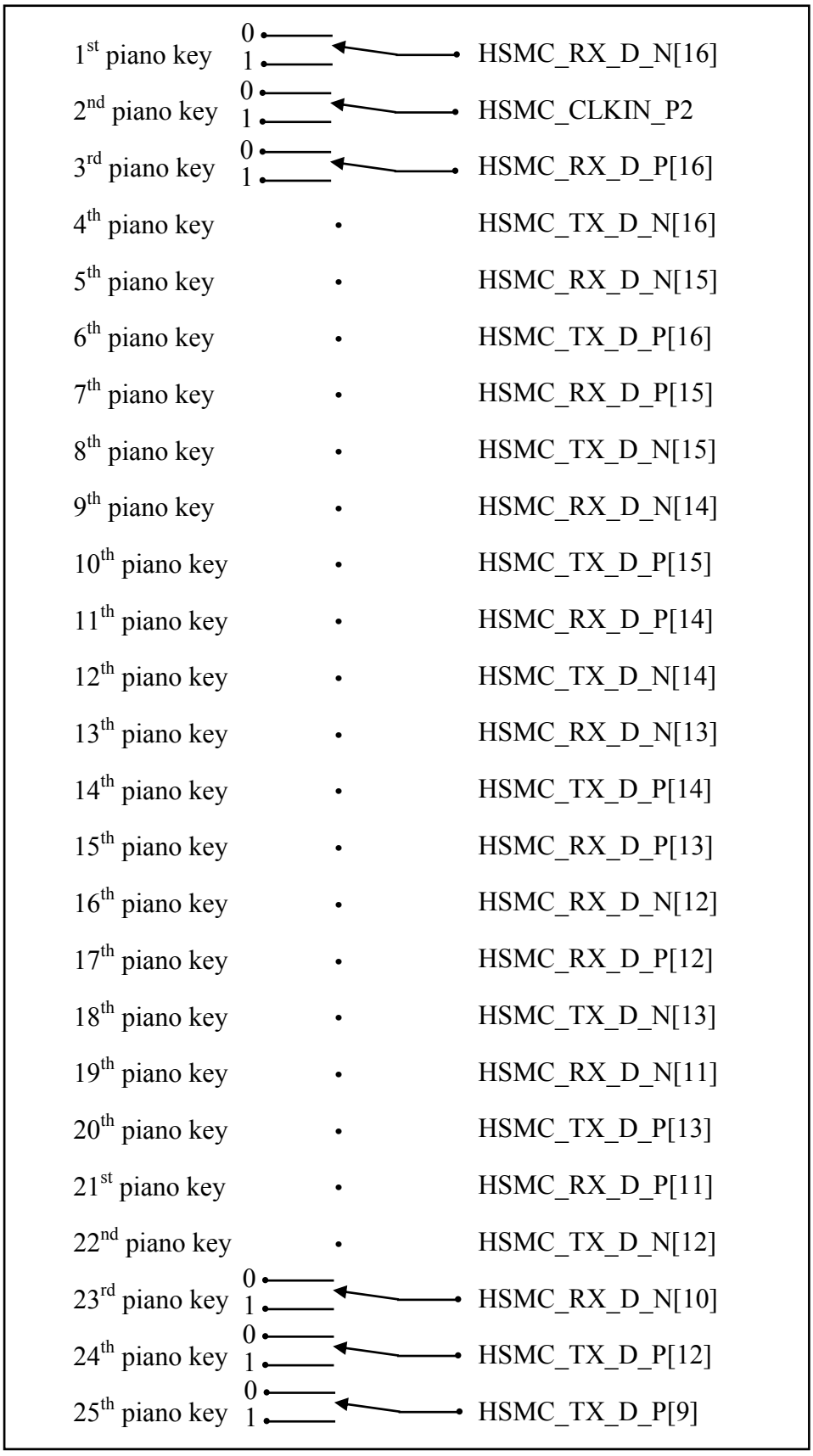
REFERENCES

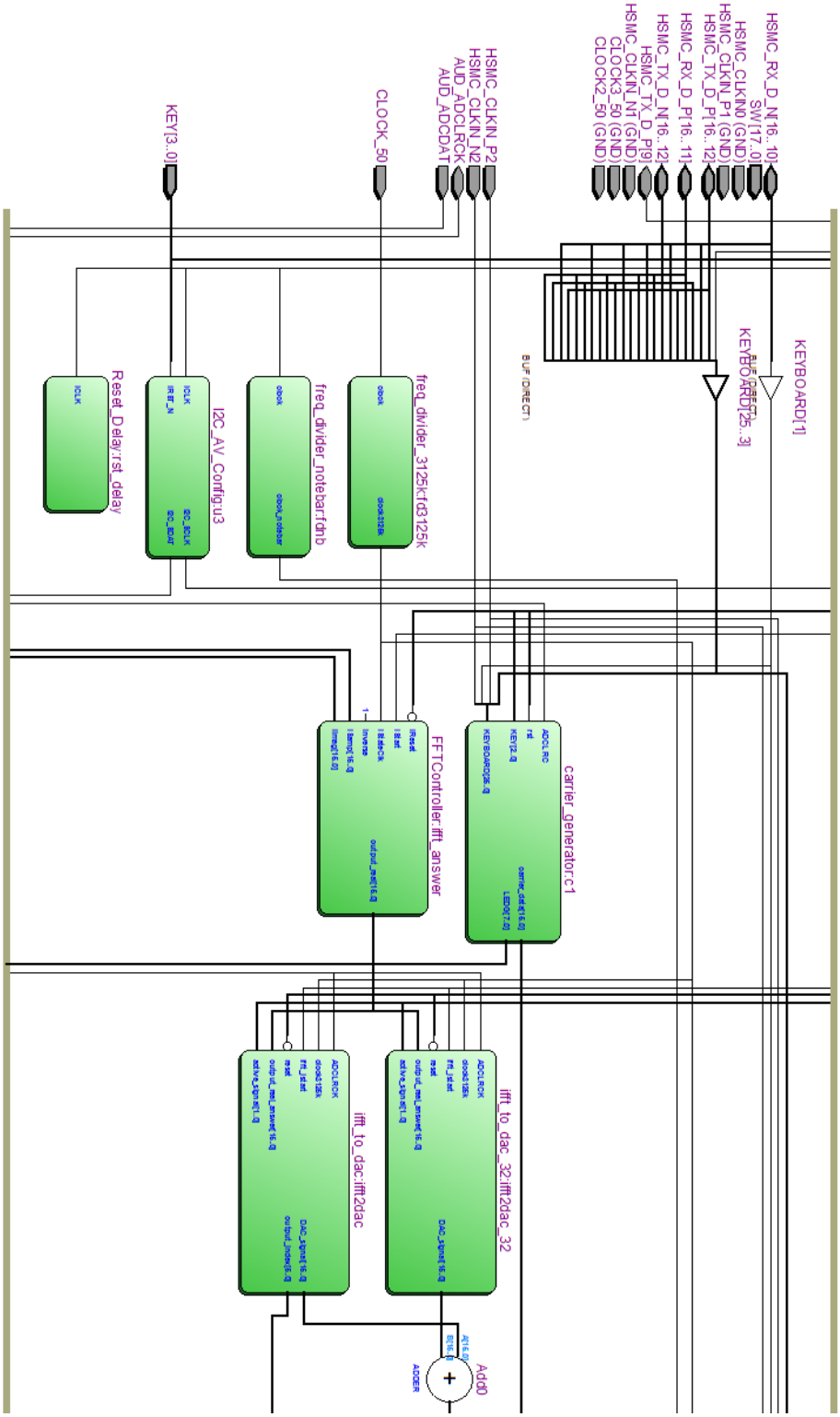
[1] B. Gold, C. M. Rader , “The Channel Vocoder,” *IEEE Transactions on Audio Akd Electroacoustics,* vol. –4C–15, ko. 4, Dec. 1967.

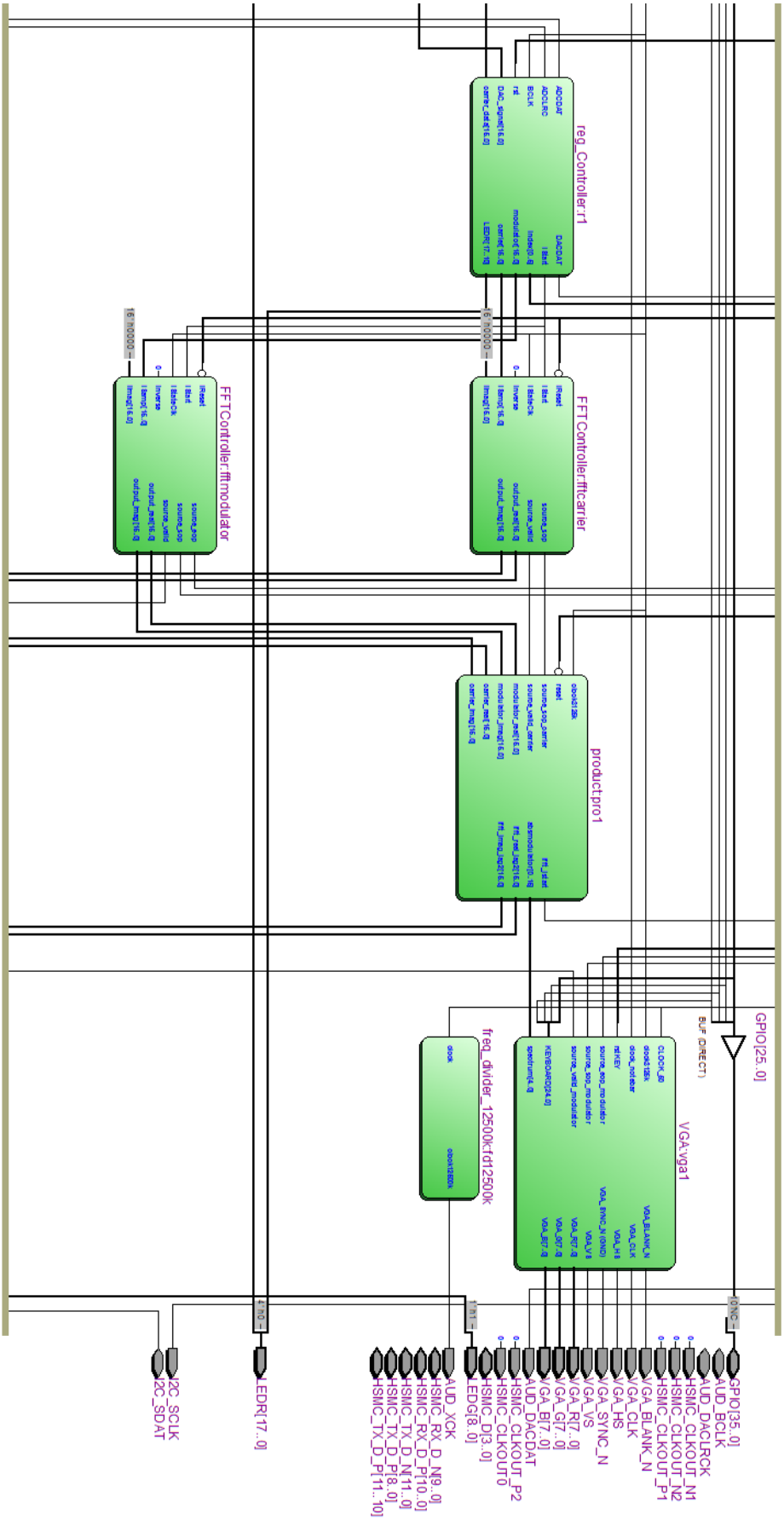
[2] D. Rocchesso, *DAFX: Digital Audio Effects*, Wiley Online Library, 2002.

[3] http://sethares.engr.wisc.edu/vocoders/channelvocoder.html

APPENDIX


1st piano key
2nd piano key
3rd piano key
4th piano key
5th piano key
6th piano key
7th piano key
8th piano key
9th piano key
10th piano key
11th piano key
12th piano key
13th piano key
14th piano key
15th piano key
16th piano key
17th piano key
18th piano key
19th piano key
20th piano key
21st piano key
22nd piano key
23rd piano key
24th piano key
25th piano key
0
1
HSMC_RX_D_N[16]
HSMC_CLKIN_P2
HSMC_RX_D_P[16]
HSMC_TX_D_N[16]
HSMC_RX_D_N[15]
HSMC_TX_D_P[16]
HSMC_RX_D_P[15]
HSMC_TX_D_N[15]
HSMC_RX_D_N[14]
HSMC_TX_D_P[15]
HSMC_RX_D_P[14]
HSMC_TX_D_N[14]
HSMC_RX_D_N[13]
HSMC_TX_D_P[14]
HSMC_RX_D_P[13]
HSMC_RX_D_N[12]
HSMC_RX_D_P[12]
HSMC_TX_D_N[13]
HSMC_RX_D_N[11]
HSMC_TX_D_P[13]
HSMC_RX_D_P[11]
HSMC_TX_D_N[12]
HSMC_RX_D_N[10]
HSMC_TX_D_P[12]
HSMC_TX_D_P[9]


HSMC_RX_D_N[16..10]
SW[17..0]
HSMC_CLKIN0 (GND)
HSMC_CLKIN_P1 (GND)
HSMC_TX_D_P[16..12]
HSMC_RX_D_P[16..11]
HSMC_TX_D_N[16..12]
HSMC_CLKIN_N1 (GND)
HSMC_TX_D_P[9]
CLOCK3_50 (GND)
CLOCK2_50 (GND)
KEY[3..0]
CLOCK_50
HSMC_CLKIN_P2
HSMC_CLKIN_N2
AUD_ADCLRCK
AUD_ADCDAT
KEYBOARD[25..3]
KEYBOARD[1]
BUF (DIRECT)
Reset_Delay:rst_delay
I2C_AV_Config:u3
freq_divider_notebarfdnb
freq_divider_312Kfd3125k
FFTController:fftt_answer
carrier_generatorcc1
ifft_to_dac:ifft2dac
ifft_to_dac_32:ifft2dac_32
Add0


reg_Controller:r1
FFTController:fftcarrier
FFTController:fftmodulator
productpro1
freq_divider_12500k:fd12500k
VGA:vga1
GPIO[25..0]
GPIO[35..0]
AUD_BCLK
AUD_DACLRCK
HSMC_CLKOUT_N1
HSMC_CLKOUT_N2
HSMC_CLKOUT_P1
VGA_BLANK_N
VGA_CLK
VGA_HS
VGA_SYNC_N
VGA_VS
VGA_R[7..0]
VGA_G[7..0]
VGA_B[7..0]
AUD_DACDAT
HSMC_CLKOUT_P2
HSMC_CLKOUT0
LEDG[8..0]
HSMC_D[3..0]
AUD_XCK
HSMC_RX_D_N[9..0]
HSMC_RX_D_P[10..0]
HSMC_TX_D_N[11..0]
HSMC_TX_D_P[8..0]
HSMC_TX_D_P[11..10]
LEDR[17..0]
I2C_SCLK
I2C_SDAT