Air Guitar on Altera DE2-70 FPGA Architecture

Ger Yang, Tzu-Ping Sung, Wei-Tze Tsai, advised by Shao-Yi Chien
Department of Electrical Engineering, National Taiwan University

Abstract — Air guitar is a well-known competition all over the world. It gives us a passion for those who might not play the guitar so well can be indulging in the magic power of guitar as well. However, although one may act so seriously as if he/she has a guitar, the sound would never be produced. Therefore, by capturing the user's movement, we proposed a real-time and equipment-free system to make air guitar not only acting but also an alternative and user-friendly way to play the guitar.

Keywords — Air Guitar, FPGA, Real-Time, Gesture Detection

I. INTRODUCTION

Worldwide air guitar competitions are regularly been held in many countries for years. An "air-guitarist" does not play a real guitar. Instead, he attracts audience by exaggerated motions pretending he is really playing something, together with pre-recorded rock or heavy metal-style music, dazzling lights, and loud singing or lip-synching. Nevertheless, as we have just mentioned, an air-guitarist can only play pre-recorded songs, and thus limiting his performance only a kind of dance. What we are going to do is to endow an air-guitarist the capability of playing a tune impromptu. Besides, we also transform the complicated fingering into simple hand gestures so that even the ones who have never learnt how to play a guitar can easily play our design.

In 2005, students from Helsinki University of Technology developed a system, which translates a performer's hand movements into sounds by recognizing a pair of gloves of specified color via video camera. Today this system is exhibiting at the Helsinki Science Center, and for more details, you may refer to [1] and [2]. Later in 2006, Australian researcher Helmer et al developed Smart Textiles [3], which can be used to sense human movement by embedding sensors within clothes. Its application to the air guitar is an "Air-guitar T-shirt". In 2007, Japanese company Takara Tony introduced a product called "Air Guitar Pro" [4], which is a small device imitating the guitar neck with numerous buttons on it. A performer takes it and selects the tones via the buttons. Besides, there are infrared motion sensors on the bottom of the device, acting as virtual strings, and a performer is able to wave his hand by the sensor to play sounds. Recently, there are also some air guitar applications on the Microsoft Kinect or Apple iOS devices [5], however their functions are limited owing to their frameworks.

We are going to implement our air guitar system on the Altera DE2-70 FPGA architecture. The performer's motion will be captured by a video camera, and then be processed by hardware-level image processing and recognition techniques, which is similar to Helsinki students' work in [1] and [2]. However, we do not require users to wear gloves with specified colors, but detect the skin color instead. Furthermore, our system is able to recognize various hand gestures and performers can easily play music of different styles precisely.

This paper will be organized as follows. First, in section II, we are going to give a detailed description to the features of our system, including a simple tutorial to play our air guitar. Then, starting from section III, we are going to illustrate the technical details, such as platform, system architecture diagrams, and the detailed algorithms applied in each functional block. In the end, we will briefly give the usage of the resources on the FPGA board and then give a conclusion to our design.

II. FUNCTION DESCRIPTION

The Air Guitar aims to provide a new interface for the user who has never played a guitar before to play a virtual guitar like real. Therefore, it is necessary to give the user an experience analogue to how a guitar is played in reality. After observing how one actually plays a guitar, we concluded that the completeness of the guitar performance composes of three parts which include finger positioning, strumming and sound generating from the body. As a result, we designed these parts individually in order to meet our requirements which we mentioned above. That is, we aim to make playing air guitar like real.

The main components are gesture detector, motion detector and audio synthesizer corresponding to finger positioning, strumming and sound generating, respectively. We will give a brief introduction to them as following:

A. Gesture Detection

Guitar is so popular for its ability to play plentiful chords and notes for an accompaniment. The basis of a guitar to be diversifying is the six chords on the neck. The fingers' position for each chord gives a variety of chords. However, it is hard for new comers to memorize so many chords and position the fingers correctly. Therefore, we simplified the gestures so that everyone can play Air Guitar in a very simple way. We support up to twelve different gestures which represent to twelve kinds of chords and notes in Figure 1 (a). To detect the gesture, we may put our left hand in a specified detection area where we recognize which gesture it is.

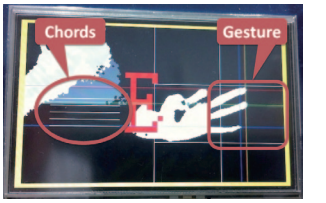

Figure 1. left (a) and right (b)

B. Motion Detection

The most fascinating part when playing a guitar is strumming through the chords and making a harmonious sound. There are 5 thin detection bars which are shown in Figure 1 (b) for our right hand to strum through. Therefore, we may generate sound through 5 detectors and each simulates an independent chord. The sound from them then composed to be a chord, e.g. C major.

C. Audio Synthesizer

It takes two steps for a guitar to make sound which are the vibration of the steel chords and the resonance of the body. This results in a special sound for the guitar. Therefore, to synthesize the sound of the guitar, we need to make the amplitude curve, waveform and pitch to be similar enough as it really is. The audio synthesizer is specially designed to generate a single note of the guitar sound with different pitches. As a result, we may use it to generate a variety of chords by different combinations.

D. Instruction

We can see from Figure 1. To play with the Air Guitar, we simply need both of our hand, one to change gesture and the other to strum the

chords. As show in the display panel, there are two regions. The gesture detector is located in the right area and the motion detector is in the left area with five chords. We need to change our left hand gesture according to Figure 1 to get different kinds of chords and in the meantime strumming our right hand on the other region to get sound. In addition, we may choose single note mode and chords mode to play with more fun. From description above, we have proposed a guitar-free performing system which gives the same user experience as playing it real.

III. ENVIRONMENT & COMPONENTS

Our system is based on the Altera Cyclone II EP2C70 FPGA built on the DE2-70 board. A 500-megapixel D5M camera with 2560×2160 full-resolution is used to capture the performer's motion, and a LTM LCD is for the display with the SDRAM on the DE2-70 board. Besides, the synthesized audio is output to a speaker via the DAC on the DE2-70 board. In addition, in order to make our system work more accurately, sometimes we need some fill lights when the environment light source is not homogeneous.

Figure 2 is the compilation result of our implemen-tation. The compiler we used is Quartus II 10.0. About 21% of logic elements on the FPGA are used.

Flow Status	Successful - Tue Aug 21 19:52:00 2012
Quartus II Version	10.0 Build 218 06/27/2010 SJ Full Version
Revision Name	DE2_70
Top-level Entity Name	DE2_70
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	No
◢ Total logic elements	14,559 / 68,416 (21 %)
Total combinational functions	13,836 / 68,416 (20 %)
Dedicated logic registers	2,781 / 68,416 (4 %)
Total registers	2819
Total pins	530 / 622 (85 %)
Total virtual pins	0
Total memory bits	131,240 / 1,152,000 (11 %)
Embedded Multiplier 9-bit elements	41 / 300 (14 %)
Total PLLs	2 / 4 (50 %)

Figure 2.

IV. SYSTEM ARCHITECTURE

In this section, we are going to introduce the design architecture of our system. Figure 3 (a) is the data flow diagram, which provides a simple view to our design. Figure 3 (b) is the detailed system diagram, in which the relationship between each functional block is shown. Our system is composed of three stages, the Pre-processing Stage, the Detection Stage, and the Synthesizing Stage. We are going to give a brief introduction to each stage in the following part.

A. Pre-processing Stage

In this stage, the video signals are being processed to the format that can be dealt with in the next stage. First, the video frames are captured from the D5M camera and down-sampled to 800×600 resolution with 30-bit RGB color space. Then, the video frames are being sent to the skin detector. In the skin detector, heuristics are applied for determining the skin color parts on each frame. After that, the result is filtered to remove noises and interferences so that the detectors in the next stage are able to make their decision based on the clear skin color information.

B. Detection Stage

In this stage, there are various kinds of detectors to figure out what kind of moves the performer is doing. A motion detector is a functional block that detects whether the performer is waving his right hand over the virtual strings. Each virtual string is implemented as an independent module. A gesture detector recognizes the performer's gesture of his left hand, and is composed of four parts: finger detector, palm detector, thumb detector, and a decision agent that collects the information from the previous three parts. Next, their results are sent to the next stage, the synthesizing stage.

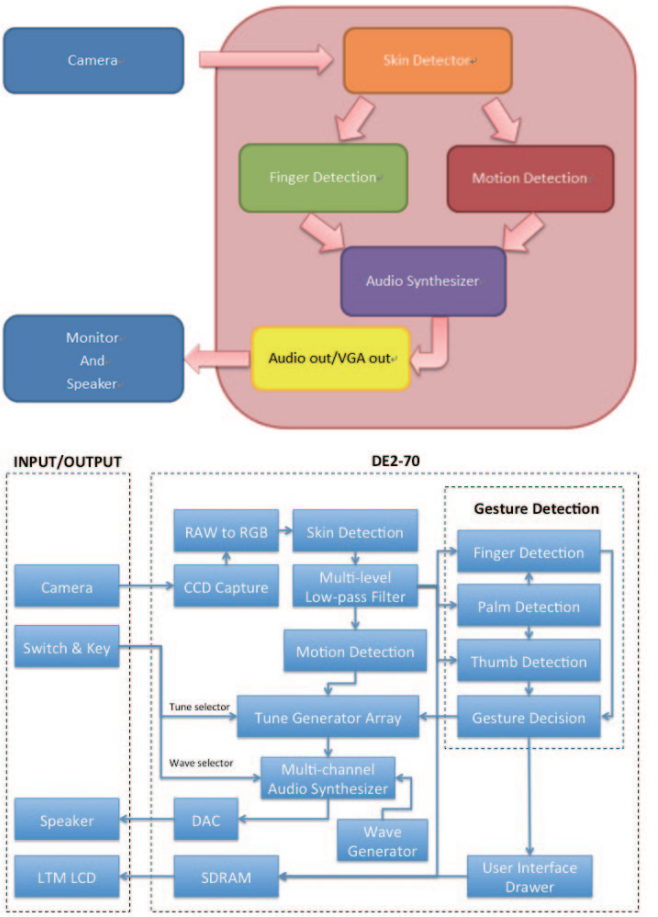

Figure 3. top (a) and bottom (b)

C. Synthesizing Stage

The audio synthesizer generates the tones and output via the DAC and then to the speaker. Just as we have mentioned in the previous sub-section, each string is implemented as an independent module. As long as the user waves his hand over a virtual string, the corresponding motion detector attracts it and then triggers a tune generator. A tune generator will decide the physical properties according to the information given by the gesture detector. After that, the module audio synthesizer collects the information from the tune generator array and then synthesizes the sounds.

V. SKIN DETECTION ALGORITHMS

A. Skin Detection Heuristic

The video frames captured from a D5M camera is firstly transformed into 30-bit RGB color space, in which 10 bits ranging from 0 to 1023 represent each color. In such a color space, we follow the heuristic rule to determine whether a pixel is of skin color.

$$R > 380 \;\&\; G > 160 \;\&\; B > 80 \;\&$$
$$R > G \;\&\; R > B \;\&\; R - G > 60 \;\&$$
$$R - B > 60 \;\&\; (B < 550 \mid R - B > 320)$$

The heuristic is based on [6] and is modified to adapt the skin color of Asian.

B. Filtering

However, we found that with only this heuristic, the noises and interferences will deeply affect the precision of the gesture detector. We apply a filtering scheme, which we call it "multi-level low-pass filter", to eliminate its effect.

The filtering scheme is composed of many levels. In each level, every pixel in a given frame is compared with its eight adjacent pixels which is shown in Figure 4. If there are more than six of them are of skin color, then the pixel is claimed as a skin-colored pixel. The reason for adopting this scheme is that, we find out that pixels of skin color detected from real skin tends to be gathered as a cluster, while skin color pixels from noises and interferences are usually separated as small pieces.

The process mentioned in the above part is done for three times. According to our experiment, repeating the process for three times can filter out about 80% of noise and interference in most cases. And this is the reason that we call this process the "multi-level" filtering.

+1	+1	+1
+1	0	+1
+1	+1	+1

Figure 4.

Figure 5. left (a), middle (b), and right (c)

VI. GESTURE DETECTION ALGORITHMS

The gesture detection algorithm we designed is real-time and non-buffered. Therefore, the algorithm responses fast and it requires no extra memory. The gesture detector is composed of four minor building blocks, which are palm detection, finger detection, thumb detection, and gesture decision. In the gesture decision block, it will collect the information generated by the previous three parts and then make a final decision of which gesture the user is making.

A. Palm Detection

The main purpose of the palm detector is to identify the upper and bottom boundary of a palm. Let us first learn how the video signal is read in. The video signal is read sequentially from left to right as a line, and up to down line by line. Thus, we count the number of skin color pixels in each line. By heuristic, the palm area is defined as the lines with more than 80 skin color pixels. Hence, we are able to find the upper boundary of the palm, which is the first line of the palm area that exceeds the threshold, and we may find the upper y-coordinate of palm. Similarly, the bottom boundary, which is the last line of the palm area, is the lower y-coordinate. The result is shown Figure 5 (a).

B. Finger Detection

The work to detect finger gesture is a relatively hard work. However, for the gesture we defined, they are particularly in the same direction as we showed previously in Figure 1 (a) and the difference between each gesture is basically the number and the presence of fingers. Therefore, we may add the constraint of hand orientation to make the detection in a predictable way. Moreover, we try to reduce the problem to fingertip

identification. The finger detection algorithm has two steps, interest peak identification and fingertip detection policy. In the first step, we are trying to find out all the possible candidates of fingertips. And in the second step, we try to rule out all the misclassified fingertips.

To achieve the first step, we first find the rightmost (largest x-coordinate) skin color pixel in each scanning line. Next, we will compare these rightmost pixels to find interest peaks. The peaks are illustrated in Figure 5 (b). In this step, we may observe that there are still many misclassified fingertips.

In the second step, we set up a policy with two rules to eliminate the wrong fingertips. The first rule is that the difference of y-coordinate between identified peaks must exceed 15 pixels. That is, two peaks which is close in the y-direction should only be counted once since there exists a least distance between two fingers. The final rule is that fingers should have similar x-coordinates except for the little finger when the hand is oriented in the desirable way. As a result, we may see that misclassified peaks are eliminated in this case. The result is shown in Figure 5 (c).

C. Thumb Detection

The purpose that we designed an individual detector for thumb without detecting it in the finger detection is that thumb has a distinct orientation compared to the others. Therefore, we develop another way to deal with thumb. Contrary to finding peaks, thumb detection is trying to count the number of skin color pixels in the thumb area which is the area above the upper boundary of the palm we've been detecting continuously. If the number exceeds a threshold, the thumb is said to be raised. Together with the finger detection, we may support 12 different gestures.

D. Gesture Decision

From the information above we may make a decision for the gesture. The resulting gestures are shown in Figure 1 (a) and the supporting chords and tones are shown in Figure 6.

Chords	Tones	Chords	Tones
C	B.	A	C7
D	C	B	Dm
E7	D	C˙	Em
F	E	D˙	Fmaj7
G	F	E˙	G7
Bm	G	F˙	Am

Figure 6.

VII. MOTION DETECTION ALGORITHMS

To detect the motion of hands, we don't need too much information of the shape and gesture. All we need to do is to detect well on whether a hand actually strum through it. That is, we need each motion to be detected rapidly. On contrary, the detector should detect nothing when there is no hand over it. To meet the requirement we designed a module which checks a specific region at each frame. When the number of the filtered skin color pixels in the region exceeds a threshold, the region is set to be in "covered" state. On the other hand, as the number doesn't the threshold, it is set to be in "uncovered" state. Furthermore, the sound is played if the state transition satisfies from "uncovered" to "covered." The reason is that it is accordance to a chord is really plucked. In addition, the threshold is acquired in a heuristic way. Moreover, the process is non-buffered since we check each horizontal line in each scanning. As a result, the reaction time is rapid and can be said to be real-time.

VIII. AUDIO SYNTHESIZER

We generate the sounds by using a synthesizer instead of recording them in advance. In order to simulate the sounds of a guitar, we generate them according to their physical properties, the timbre, pitch, and the amplitude curve. Besides, each virtual string is implemented as an independent module, which is called a tune generator, and the physical properties are adjusted in it.

A. Timbre & Pitch

Physically, to generate a sound, we have to know its timbre and pitch, or waveform and frequency, in academic terms. Given a waveform in a period, theoretically we are able to generate the sound of any frequency. In digital systems, a waveform is represented as a series of samples over a discrete-time scale. Consider such a periodic waveform of frequency f, given the sampling period T_s, the phase difference between consecutive samples is $2\pi fT_s$. Suppose there are N samples in a period, and we are going to generate a sound of frequency f, we must play the waveform within the time T=1/f, and thus we have a sound with sampling frequency Nf. This means that to play the generated sound with the sampling frequency $1/T_s$, we can just simply down-sample the sound with the sampling frequency Nf.

In our system, we describe the waveform as 64 samples and store them in the module wave generator. Though there are some errors owing to the implementation limitation, the pitch of each tone has been carefully tuned by a tuner, and is made sure that the chords sound harmonically to human ears.

B. Amplitude Curve

According to [7] Curve, source not found], the amplitude curve of a guitar sound can be modelled as exp(k/t), as shown in Figure 7. Further, we can approximate it into two linear parts. The first part is about the 0.3 seconds from the beginning, in which the amplitude diminishes really fast, where the second part lasts from the 0.3 second and decrease on a relatively slow rate. At the end of the first part, the volume is about only one-eighth compared to that in the beginning. We follow this rule to generate the sound. When a second sound is played before the first sound ends, the amplitude curve is reset to the beginning state, which means that a new waveform will overlay the original one.

C. Mixer

There are five virtual strings in our system. Since each string is implemented independently, as we have mentioned in the previous sections, we need a mixer to integrate the five channels. The mixer formula is designed with some considerations. First, we must avoid overflows. Second, owing to the characteristics of human ears, the sounds with higher frequency tend to be louder even if the volumes are identical, they must be weakened. Suppose $\{S_k\}$ is the strength of the sounds in the five channels, on the increasing order of the frequency. The mixer formula is given as the following heuristic:

$$m = S_1 + \tfrac{3}{4} S_2 + \tfrac{1}{2} S_3 + 3/8 (S_4 + S_5)$$

You may notice that this formula is not normalized. In fact, the result is right-shifted so that overflow would not occur.

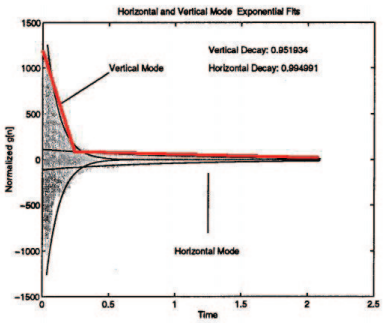

Figure 7.

CONCLUSIONS

With all the detection techniques discussed above and the synthesis of guitar sound, we proposed a real-time, non-buffered and equipment-free Air Guitar system to give users a new experience playing the guitar with bare hand and easy gestures.

ACKNOWLEDGMENT

We wish to acknowledge Altera® Inc. and Terasic® Inc. for their sponsor and hard works holding the competition which provides us with a chance to learn from the project. In addition, we also wish to acknowledge Professor Shao-Yi Chien and National Taiwan University for providing us with a very good environment and training.

REFERENCES

[1]. Teemu Ma "ki-Patola, Juha Laitinen, Aki Kanerva, and Tapio Takala. "Experiments with virtual reality instru- ments". In Proceedings of the 2005 conference on New interfaces for musical expression, NIME '05, pages 11- 16, Singapore, Singapore, 2005. National University of Singapore.

[2]. Teemu; Kanerva Aki; Huovilainen Antti Karjalainen, Matti; Ma "ki-patola. "Virtual air guitar". J. Audio Eng. Soc, 54(10):964-980, 2006.

[3]. R.J.N. Helmer, M.A. Mestrovic, D. Farrow, S. Lucas, and W. Spratford. "Smart textiles: Position and motion sensing for sport, entertainment and rehabilitation." Advances in Science and Technology, 60:144-153, 2009.

[4]. [Online]: http://www.foxnews.com/story/0,2933,287115,00.html

[5]. Hamed Ketabdar, Hengwei Chang, Peyman Moghadam, Mehran Roshandel, and Babak Naderi. "Magiguitar: a guitar that is played in air!" In Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services companion, MobileHCI '12, pages 181-184, New York, NY, USA, 2012. ACM.

[6]. V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In Proc. Graphicon, volume 3, pages 85-92. Moscow, Russia, 2003.

[7]. SK. Bradley, M. Cheng, and V.L. Stonick, "Automated analysis and computationally efficient synthesis of acoustic guitar strings and body," in IEEE ASSP Workshop on Applications of Signal Processing to Audionand Acoustics, New Platz, NY, October 1995.