Pah! Pah! Pah!

A voice activated video game

Hao Hu(hh2506), Kezhen Liu(kl2688), Shaobo He(sh3156), Sheng Zheng(sz2372), Yi Su(ys2646)
Short name: Pah!

Abstract

The goal of our project is to port a voice activated iPhone game called Pah to Altera DE2 Board. This proposal will provide a rudimentary idea of implementation of this game, i.e. the jobs of hardware and software. Milestones of this project will also be presented.

Overview of Project

Pah! is a very popular game on the iOS platform which every member of our team really enjoys. So we decide to port this game to a FPGA board for three essential components of this game, voice processing, computing core, screen display are provided by DE2 Board. The prototype of this game is a spacecraft either dodging meteoric stones by doing up and down or destroying them by launching missiles. Both of the two actions are controlled by voice, namely when the tone of "Ah" goes high, the spacecraft goes upper and when the player issues "Pah", one missile is sent out. The spacecraft goes down by default. When it hits the meteoric stone, game is over. Our plan is to implement the first function while leaves the second as an option depending on our schedule.

What it looks like

Since this game is migrated, it behaves very similar to its prototype on iPhone. First, voice of the player is sampled by a microphone connect to DE2 board. Then the spacecraft displayed in VGA will go either up or down depending on the trend of the player's tone.

How it may work

This section is presented by a data-flow order. The first part is voice processing. After the voice is processed, a two-bit signal, which indicates the trend of pitch, will be available to be read by the game program and moves will be done based on the value of the signal. Finally, the screen will display according to what move has been caused by the player.

Voice processing:

The first thing is to consider the processing of human voice. Since different people have different pitch of their voices, a fixed standard in spectrum will be invalid. The variation of frequency across different time is our solution for the above problem.

The FFT algorithm is a fast and effective way to get the spectrum of the voice signal. The FFT algorithm achieves its efficiency gains by decomposing the DFT into a number of smaller DFTs and exploiting the symmetry and periodicity of the sub stages to reduce the number of calculations. An n-point FFT only requires n×log2n complex multiplications. Cutting down the number of complex multiplications improves the FFT performance, often by several orders of magnitude, depending on the order of the transform. Using the FFT IP core in MegaWizard to implement the FFT in hardware will further improve the speed of spectrum analysis compared with software implementation in the Nios system.

Control generation:

The trend of the pitch is regarded as the control signal. It only requires a rough spectrum variance across time without considering the details of the spectrum. After the Fourier transform of the signal was calculated by the FFT IP core, the spectrum information should be further processed. The average frequency of some spectrum with largest amplitude will be the regarded

as the rough frequency for analyzing the trend.

HW/SW job allocation:

1.What hardware will do
Hardware part of this project is responsible for voice processing and offering hardware infrastructure like Nios2 core, avalon bus and screen. Voice process will be implemented by VHDL codes and function as a peripheral.

2.What software will do
We will write a software program just as what Pah! developer does. The only difference is that we have a dedicated voice processing component and we use C or C++.

Milestones:

Milestone 1 (April 2):

- Make detailed hardware design specification
- Interface voice processing module to avalon bus
- Configure VGA interface
- Build software skeleton

Milestone 2 (April 16):

- Implement voice processing module and validate it by writing testbench
- Build the game program and test it with designated stimulus

Milestone 3 (April 30):

- Put all together by connecting voice processing module to its interface to avalon bus
- Debug until it totally functions correctly

References:

1. FFT wiki link: http://en.wikipedia.org/wiki/Fast_Fourier_transform
2. Pah! 2.0 App Store link: https://itunes.apple.com/us/app/pah!- 2.0/id547013723?mt=8
3. Altera.com: Accelerating Nios II Systems with the C2H Compiler Tutorial