Ah-Ah-Piu
Final Project Presentation
Spring 2013 CSEE4840 Embedded System Design Final Project

Hongsen Yu (hy2340)  Ji Pei (jp3242)
Xiaolong Jiang (xj2137)  Nan Li (nl2411)
Junlin Lu (jl3925)
Content

• Overview
• Architecture
• Hardware (Audio, VGA & LED, Memory)
• Software
• Lesson Learned
**Description:** Ah-Ah-Piu is a VGA displayed voice-controlled single-player platform video game.

**Game Logic:** The player’s voice captured by the microphone controls the character in the game dodging and destroying the incoming enemies, in order to keep alive as long as possible and earn more points.
Let’s try it out
Architecture

NIOS II Processor

Avalon switch fabric

- On-chip Memory
- SRAM Controller
- Audio Controller
- VGA Controller
- LED Controller
- SDRAM Controller
- SRAM Chip
- WM8731 Audio Codec
- VGA Monitor
- LEDs
- SDRAM Chip
- MIC
Audio

Wolfson WM8731 Audio Codec

It is a programmable stereo Codec device, contains two pairs of ADC and DAC to accommodate both left and right channels of stereo audio. The convertors support 16-32 bits resolution and 8-96 kHz sampling rate. The codec is programmable via 2-wire i2c control bus which initialize the Codec’s 11 control registers from the control interface to set up the chip configuration. And the Codec communicate with the outside world through the digital audio interface.
Hardware

WM8731 Audio Codec
Audio

**Input:** Capture the player’s voice with the microphone, transform this signal into audio sound byte which represent the magnitude of the sound and provide it to the software.

**Output:** Play the sound effect and background music from the headphone.
Input

- The Codec perform the ADC on a 8kHz sampling rate and 16 bits resolution.
- Take the outputted digital data from ADCDAT signal, use a module to de-serialize the raw single-bit bit stream into 16 bits .WAV format.
- In the module we spilt external system generated reference clock (25MHz) to 8kHz and 128kHz, load every consecutive 16 bits into an array (sound byte) and with the help of audio controller, write the sound byte into one address of the register.
the software read the sound byte from the register, it will get an integer that ranging from 0 to 65,535. If the most significant bit is 0, then the integer ranges from 0 to 32,767 represents increasing magnitude. Otherwise, the integer ranges from 32,768 to 65,535 in decreasing order of magnitude.
Hardware

Audio_out
VGA

- 216-color Web color map
  
  Save storage space; Maintain high resolution.

- 4 layers

  Many elements need separate process.

- Separate control of each element

  Individual movement controlled by software
VGA

- Communicate

Buffer used in communication with software.

Fetch color map index stored in SRAM.
LED

- Directly controlled by processor

Four digitals are written to separate buffer address.

Decode and print digital value.
Hardware

Memory allocation

1. On Chip Memory
   Store the sound effect

2. 512k SRAM
   Store image data, using two projects instead of a MUX

3. 8M SDRAM
   Include Pll
Software

- Audio Input Categorization
- Collision Detection
- Game Logic
Hardware Audio

Set up the control register using the de2_i2c_av_config.v file

R2, R3:
LHPVOL and RHPVOL Volume control;
LZCEN left zero crossing detect enable;
LRHPBOTH left headphone controlling both channels

R4:
INSEL Input selected
MUTE MIC
SIDE TONE
MIC BOOST
Lesson learned

Hardware
Audio

1. Figure out the clock divider meaning in sample program, and calculate the proper clock frequency.
2. Do not code the same file in two or more programs at the same time, or you may miss some modification.
3. When doing large scale replacing of code, do it in gedit, rather than in Quartus or Nios2. The latter is much slower, and may leads to program crash.
Lesson learned

Hardware
VGA

• DSP for images
  Photoshop to convert 216-color map image.
  Matlab to extract bit data for image.

• Separate layers design
  relieve the work load of process.
  reduce compiling time.
Lesson learned

Hardware
Memory Allocation

- ROM: we should use a .mif file to describe the content in ROM.
- SRAM: Since there are some different matrix in the SRAM, we have to know the exact address and offset for each data.
- SDRAM: Due to the frequency problem, we have to add a PLL core to unify its clock with NIOS-2 CPU. Moreover, the BGM is also stored in the SDRAM.

We have created a new algorithm to calculate the correct address from SRAM to VGA.
Thanks for watching! Hope you enjoy our game!