WMT –
"Whac-A-Mole" Like Game

CSEE 4840
Embedded System Design

Lingchuan Mei: lm2908@columbia.edu
Jian Lu: jl3946@columbia.edu
Shuting Yang: sy2489@columbia.edu
Si Wang: sw2778@columbia.edu

Overview

- "Whac-A-Mole" like game system

Basic Game Logic

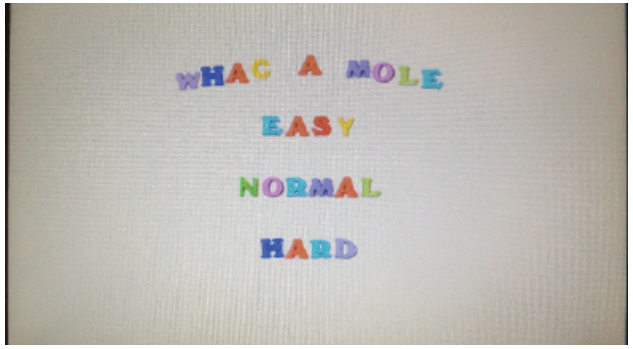
- Nine holes, Three initial lives, Two minutes
- Different items

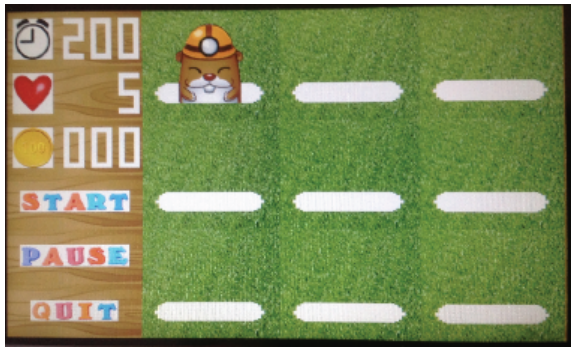


Get:	Good!	Add life	Deduct life
Miss:	Deduct life	Nothing happen	Nothing happen

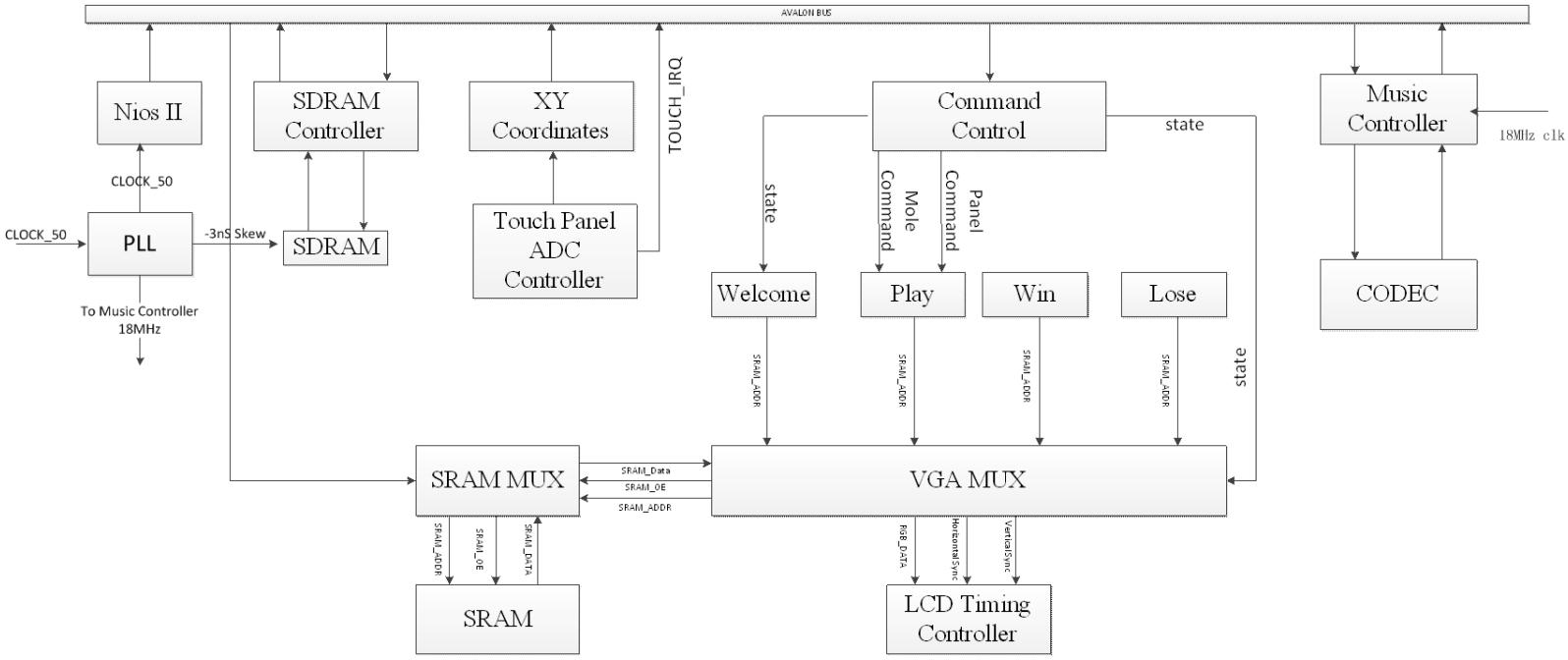
- Different levels

Basic Game Logic Cont.





System Architecture



Hardware Implementation

- Work with ADC to get the touch position x and y coordinates and read it out in Nios II
- Figure out how to display image correctly on a 800x480 screen
- Turn the x,y coordinates into useful data for software processing.
- Set up the interrupt for touching
- Figure out how to store all the data on board, basically using SDRAM and SRAM together. Using SRAM MUX.

Hardware Implementation Cont.

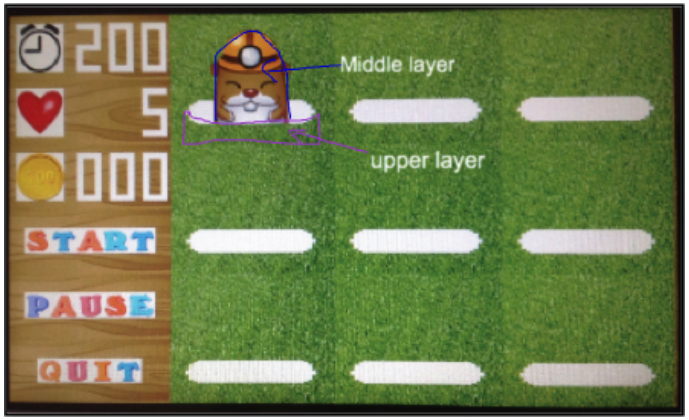
- Map all the peripherals onto Avalon bus and get rid of the ELF error
- Image conversion
- Create the state machine for the game
- Redesign a LED clock on the touch screen.

Lessons Learned About Hardware

- ELF error is usually due to wiring mistake in the top level connection of NIOS.
- Every time you copy an entire project, remember to recreate the nios project. Otherwise the BSP package is mapped to the old SOPC.
- Should have design the interface more easy to use. Should split all the variables instead of putting all of them together
- Build the design using small modules.

Display Dynamic Items

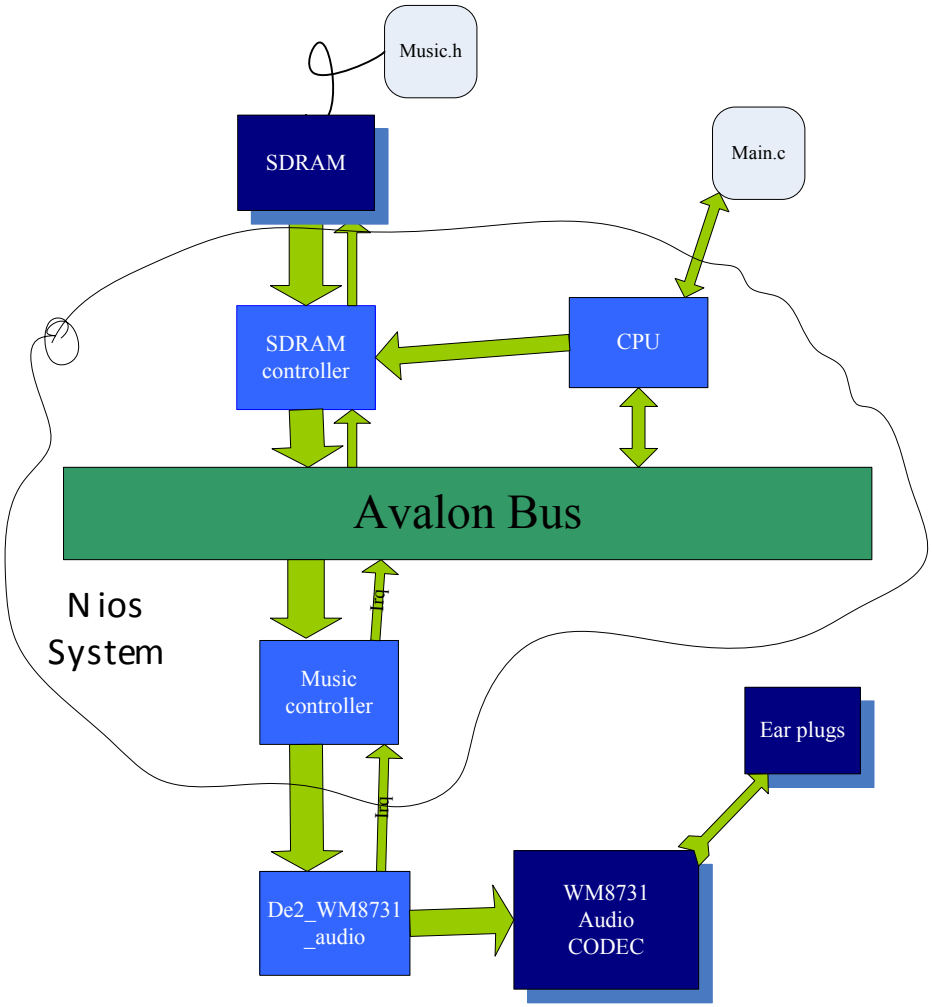
- Image -> Matrix -> Calculate the edge of the item -> SRAM
- Get the signal from the software: when, where, what
- Layers implementation



Software

- Get touch signal from hardware and identify the corresponding action
- Control the difficulty level
- Implement of the main game logic
- Keep track of the lives left, points earned and the action user take
- Send signal back to the hardware

Audio


Music.h
SDRAM
Main.c
SDRAM controller
CPU
Avalon Bus
Nios System
Music controller
Ear plugs
De2_WM8731 _audio
WM8731 Audio CODEC

Design Key Issues

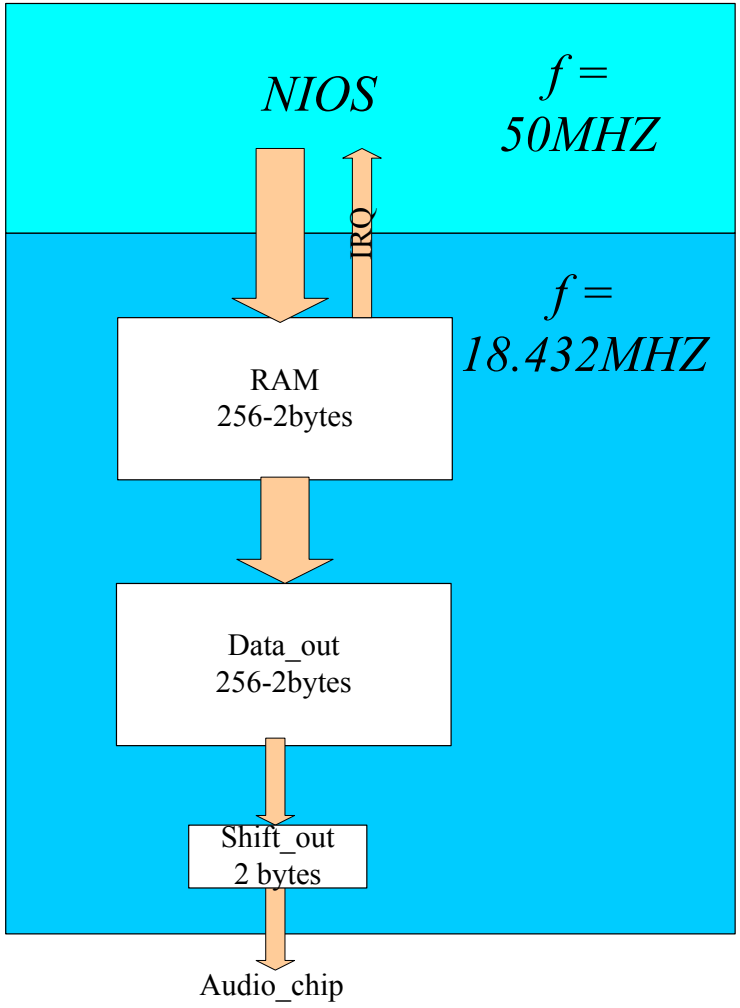
- 1.Understand how WM8731 works
- 2.Tried to synthesis music. -Too simple
- 3.Sample rates. 22050Hz. -killed 6000hz
- 4.Store music in ROM. -slow compilation

Design Key Issues

4. Data transfer

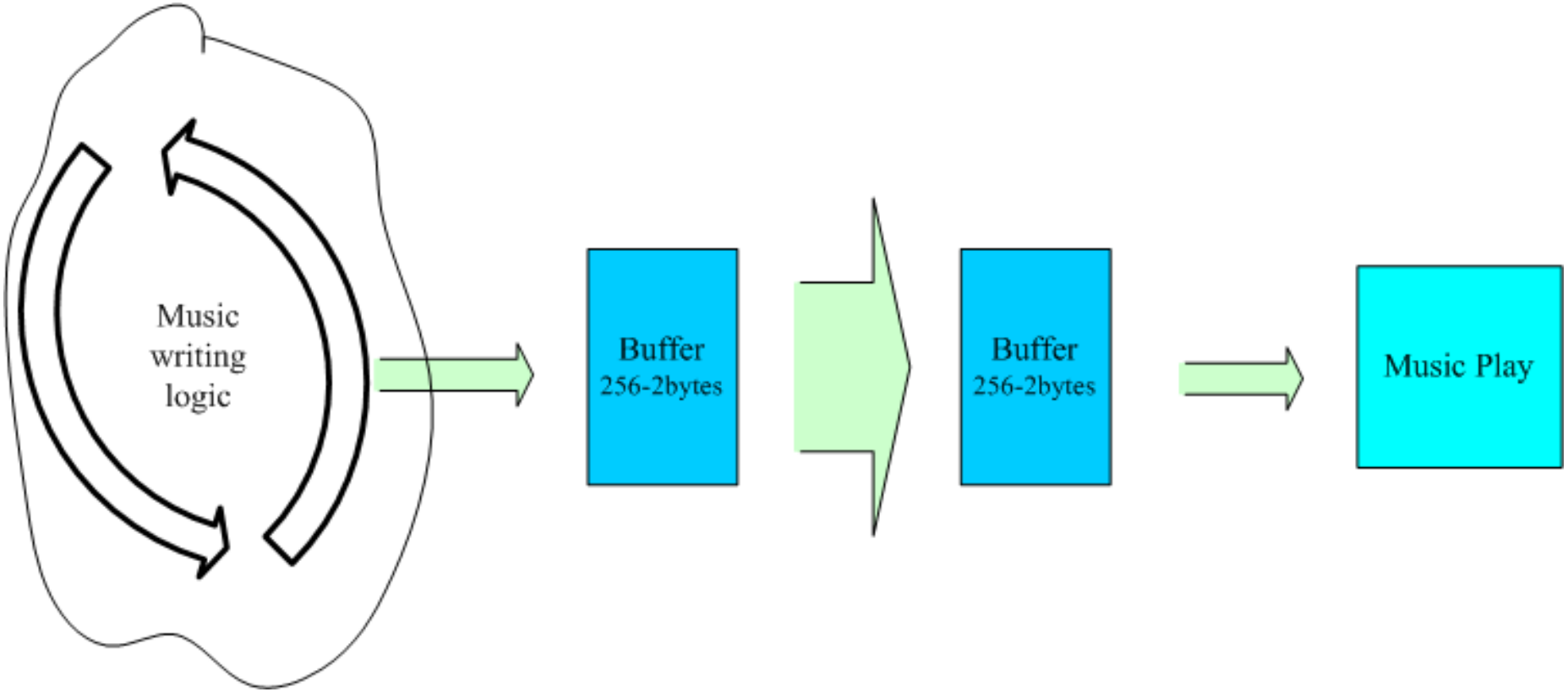
Through Avalon Bus using interrupt request.

Using buffer to buffer data transfer between clock domains



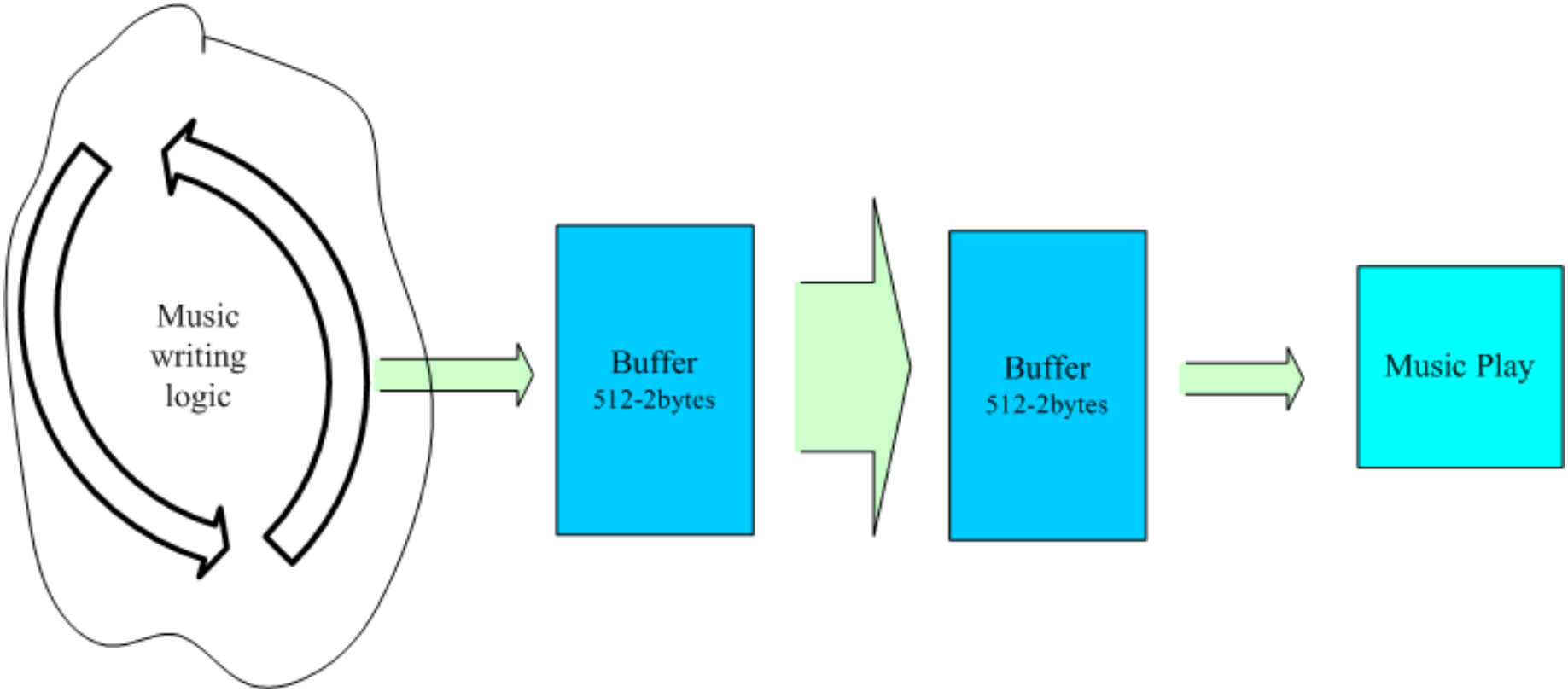
Design Key Issues

5. Structure Optimize



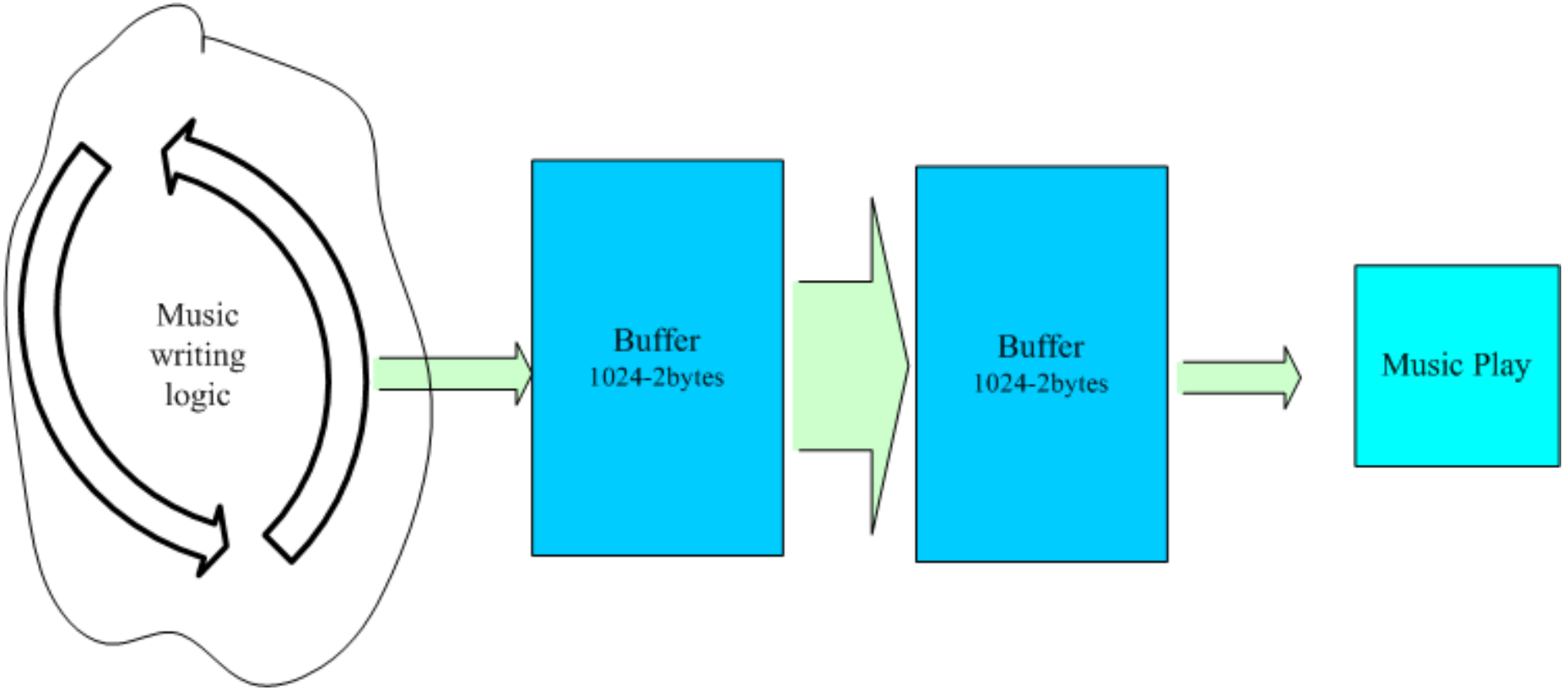
Design Key Issues

5. Structure Optimize



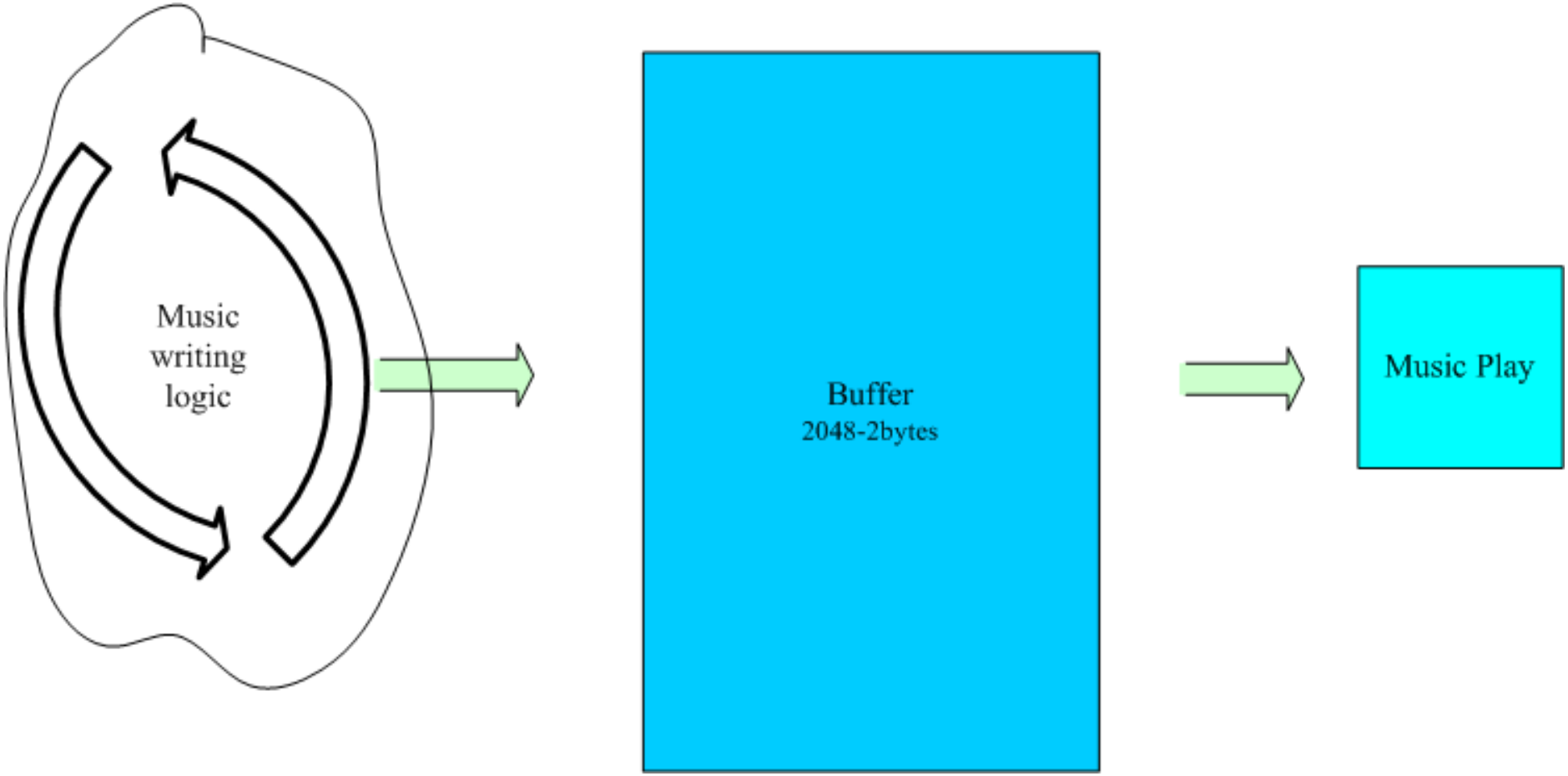
Design Key Issues

5. Structure Optimize



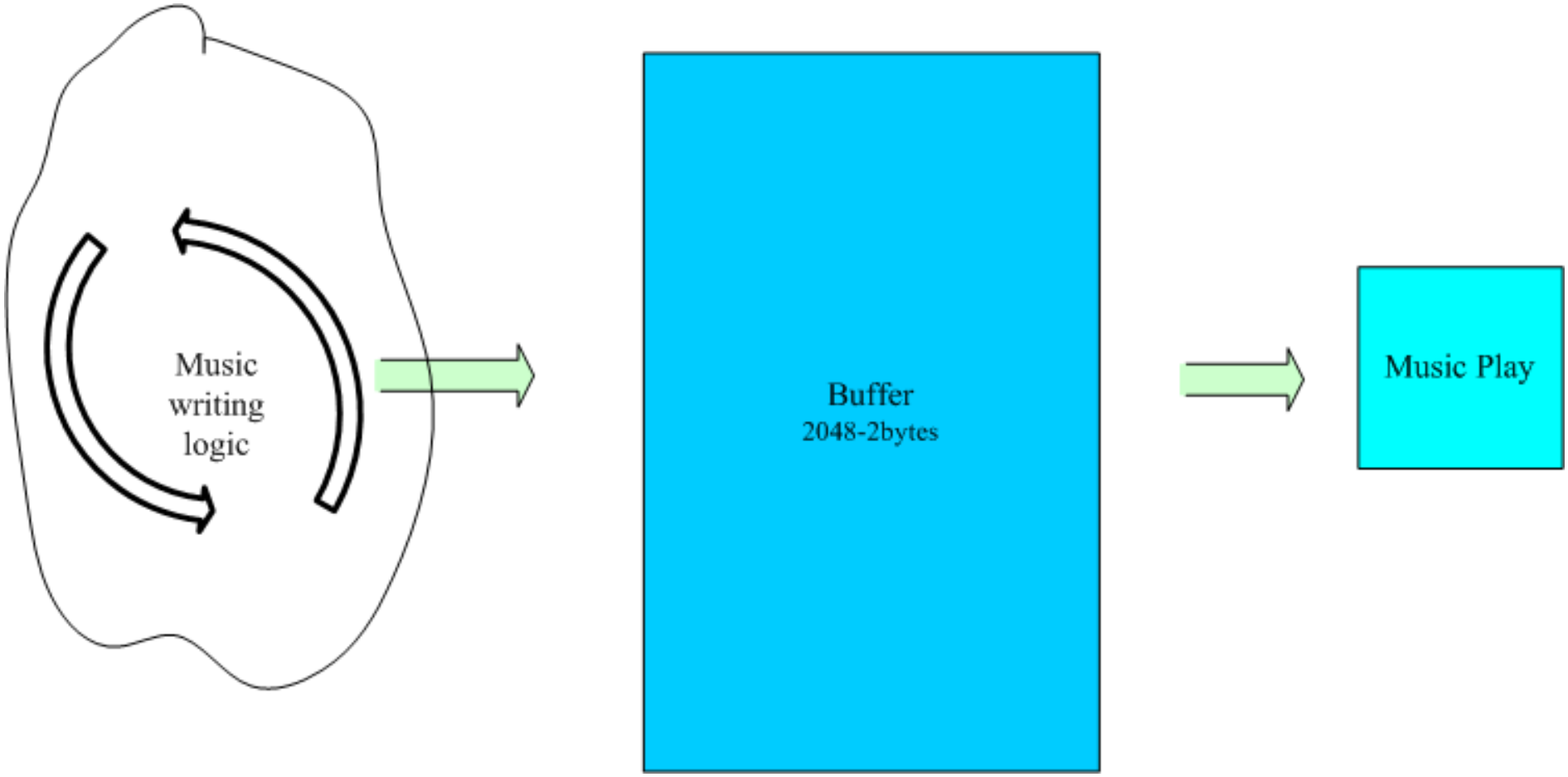
Design Key Issues

5. Structure Optimize



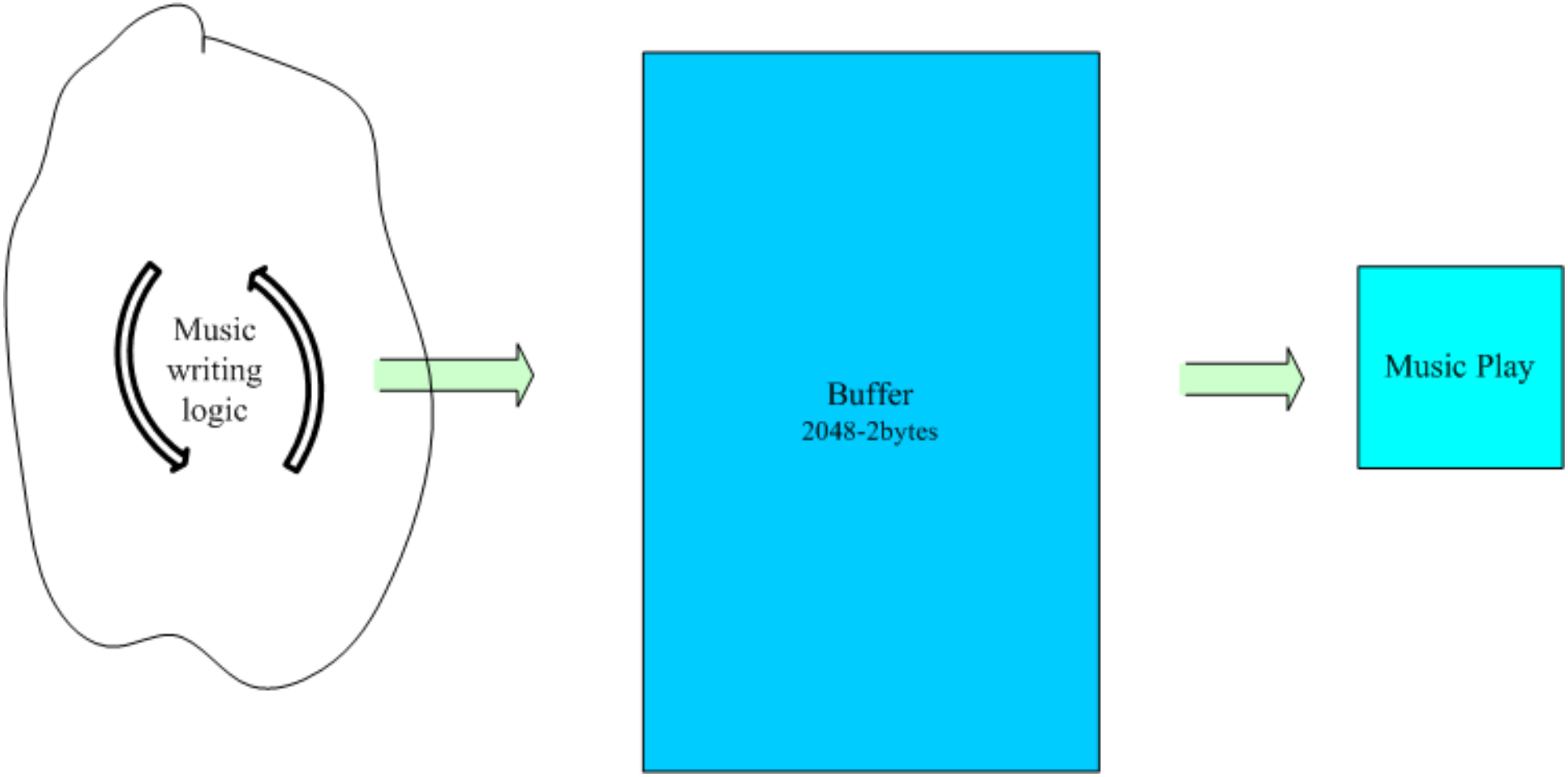
Design Key Issues

5. Structure Optimize



Design Key Issues

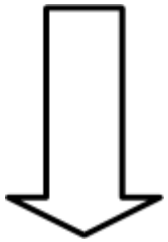
5. Structure Optimize



Design Key Issues

6. Merge music and sound effect

$$Audio = (Music + Sound)/2$$



$$Audio = Music * 1/4 + Sound * 3/4$$

Lessons learned

- Hands on experience of embedded system – interesting but time consuming
- Debug skill (both hardware and software)
- Always prepare a back up plan for big events.☺

Acknowledgement

- We WMT team would thank **dear Prof. Stephen A. Edwards and dear Professor David Lariviere** for all the professional instructions and kindly help.
- At last, we really appreciate the hard work of the TAs.

Thank you so much!

Demonstration

- Watch the video or play the game???