Speech text

* **Slide1 (Cover)**

Good evening, and welcome. We are group 18, our group members are Tang pengrui and Zheng xiaoyang.

In this project, we developed a FPGA Entertainment system that runs snake game on the Embed Fire develop board.

* **Slide2 (Table of Content)**

First, we will give a brief **introduction** of our project; then, on the **design ideas** and **overall architecture**; after that, we have a **demo video** to show in detail how the game work; and finally, we will discuss the **probem** we faced in our project, and some **conclution** to the project.

* **Slide3 (INTRODUCTION)**

First part, Introduction. You may notice that after all the stuff we talked, there is still an unexplicit point.

What is the Snake game? You may ask.

* **Slide4 (Detailed Intro)**

The **Snake Game** is a classic arcade action game where the player controls a snake to eat apples. If it eat the apple, it’s length increase, and if the snake collides with itself or the boundaries of the play area.

* **Slide5 (Feature)**

For the game to looks good, we decide our system should have following feature.

1. Tile based rendering. Allow us to render the game as tiles, allow better visual appearance.
2. We unique used Hatsuna Miku as the outfit of the Snake/main character.
3. The Snake shall gain length after eating apples.
4. Key debounce. We filtered signal of jitter input signal to get more precise control.
5. Using HDMI as output. so it’s compatible with modern devices
6. Time and Score display. To display Time and Score on the border of the screen. We used a number repository to store and allocate pixel information of number.
7. The Legend of Zelda NES fonts. We used a character repository to store and allocate pixel information of letter.

However, it is not all the points being fulfilled.

* **Slide6 (ARCHITECTURE)**

Next part, architecture.

* **Slide7 Overview**

Due to the similarity in performance of our develop board and Nintendo NES, we adopted a similar archetecture, inclulding three individual part: Cartridge system, Control system and the Visual system.

* **Slide8 (Cartridge)**

Cartridge system contains a finite state machine to decide game state, and a game logic block to run the game logic, and tells visual system what tile to render.

* **Slide9 (Visual)**

Game Render, render game pixel for graphic driver to put on screen. It contians two block: the **Sprite block** having a Sprite\_Lib that stores tile information, and can render the enetity specified by **Cartridge system**. The **Menu block** is in charge to render UI like start or end menu. And finally a mutiplexer will select which block will render the current screen.

* **Slide10 (Control)**

Control system is simple, it receive four keys’ input and filter their jitter signals, then transfer filtered input signal to key input subsection to decide direction of Snake.

* **Slide11 (IMPLEMENTATION)**

Next, we’ll play a video to show the live demo and explain how we implemente each block.

* **Slide12 (Overview)**

The demonstration video is uploaded to the YouTube page.

**VIDEO: 4 minute**

* **Slide13**

This is the IO interface we used on development board.

* **Slide14 (CONCLUSION)**

Last but not least, the conclusion section.

* **Slide15 (Reflection)**

Offline working, we did a lot of meeting to try to synchronize developing process and to communicate with groupmates to share ideas.

Division of work, we did not divide the work into parts and distribute them to members. This leads to inefficiencies of coding.

Work coordination, we did not unified writing method, or behaviour. Which is another cause of inefficiencies of working.

Design before working, we did not make a complete structure block diagram. Missing of diagram causes us having different develop thoughts.

* **Slide16 (Conclusion)**

To summarize, we design the FPGA prgram with three system to implement the snake game. And colaberate using git version controll. Finally, we successfully implemented most of our design goal.

* **Slide17 (END)**

This’s the end of our presentation, is there any questions for us.

**Video Script**

# Controller

## Key input

The controller unit read the key input and output the direction signal.

The direction is also a state machine, and it won't jump to opposide direction to avoid self collection

# Graphic Module

## VGA drawer

This module is a multiplexer to select which graphic signal to output based on current state and position on screen.

The game state determine whether to display from menu or the game.

## Tile renderer

The tile renderer take the index of current tile, and the screen position as input to generate the sprite pixel need to display at this clock.

First it convert the pixel coordinate respect to the screen to pixel coordinate respect to the grid by take the mode of grid width.

Then, it enquire from the tile data array about the pixel color value for the given tile at the coordinate,

Finally, it converts the 3-bit color into 16-bit color that readable to graphic driver.

## Display driver

The output pixel will be connect to display driver and convert to display signal to put on screen

# Game logic

## Game FSM

The game achieve it's ability to switch menu using this state machine.

It start at start menu, press a key enter the game, and if the game says game over, the game over.

Press the key again, the game start again.

## Sprite

We call the moving tiles on the game sprite, here we have the green onion, snake head, and array of snake tile.

Every individual sprite stores its coordinate, and will raise a flag if the graphic module scan to them.

## Move head

When giving a snake update clock, the head will shift one block toward the direction controller indicated.

## Shift tail

When giving a snake update signal, the coordinate of each tail will be assign to the next tail,

and the first tail will be assign to the position of head in last clock, and finish the moving process.

In addition, only tail with index smaller than tail length will cause a report of display flag when being scanned.

## Eat Apple

After the snake is moved, apple will detect if it have same coordinate as the head.

If so, it will jump to a random location, and add the snake counter by one.

## Detect Collection

After the snake is moved, game logic will check if the head have same coordinate as either a tail or the wall. If it does, it will raise the game over flag to the game FSM to indicate a end of game.