CSEE 4840 Project Design

A Tower Defense Game: **SAVE CROPS**

Team Members:

Liang Zhang (lz2460)
Ao Li (al3483)
Chenli Yuan (cy2403)
Dingyu Yao (dy2307)

Introduction:

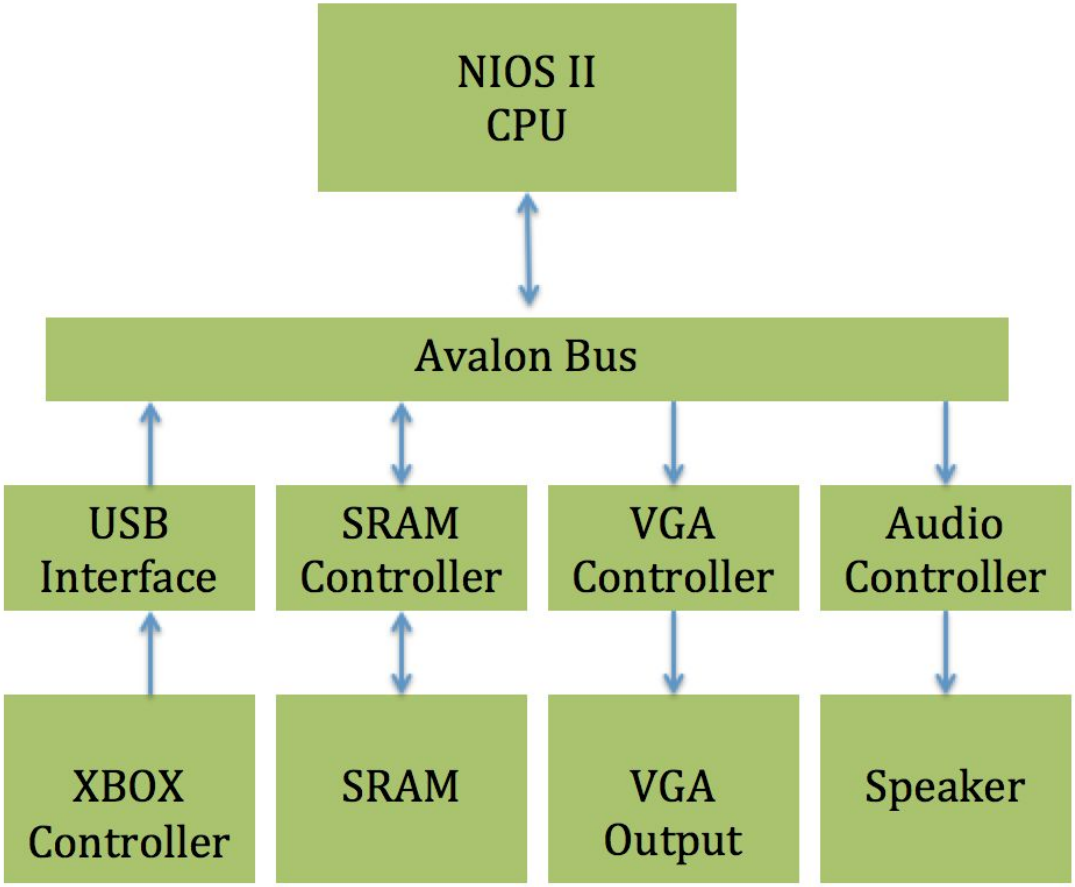
In this project, we plan to design and implement a tower defense game, named “save the crops”, which background is based on resisting several of pests. The basic original intention of the designer was that the user could make any strategy they want, and building up various kinds of defense tower besides the path that pests attacking and towers have to wipe out the pests in order to save crops. The attackers or the pests are prototyped coming from normal pests in people's daily lives, for instances fly, worms, and locusts. We design different characters with different features of appearance and characteristics, such as different life article, different attack techniques, and different defense ability. The user would get monetary rewards in each time the tower killing any pests, the system will reward user different kind of items they need automatically and randomly. By using advantage of the money rewards that system gives to user, they would call system help to large-scale destructive pests or upgrade tower characteristics for getting more powerful attacking ability. For the mapping arrangement, we are going to have single path from left most screen (source) to the right most screen (destination). The path will be looks like S type and we have path mirror that the map is bilateral symmetric. Once three attackers or pests moving from the source point to the destination point, the user is defeated, in other word, the game is over. User has to choose play again or exit. Since user getting more money rewards and reaching to specific level or passing the elementary level of the game, higher-level pests will appear to the game and user would build up higher level of defense tower correspondingly. The designer would potential let the game permanent, unless user does not want to continue the game and allow at least three attackers pass through the path to destination.

Besides the basic design idea or key points for this project, there are other key factor issues that designer need to consider and handle carefully. First is the VGA display, because map has a lot of different attackers, and towers, and other background features. We need to rewrite the driver and use different methods to solve the static and dynamic figures, such as applying sprite. Second feature point is the algorithm issue. For both attackers and defenders, designer needs to set up attackers moving along the designed path, and following different behavior and movement while they are under attack. For the defense facilities, what’s more, designer need to come up with different

characteristics based on different user level. Furthermore, what is the algorithm that defender attacking pests will also need to be considered carefully, because it directly dominates the fluency of the game and victory, defeat conditions.

Design Block Diagram:

After consideration, we initially draw a high-level block diagram as below:



As in the figure, we have a NIOS II CPU and four major peripherals: VGA display, controller, storage part and audio output module. All of above are connected to the Avalon bus so that they could communicate with the NIOS II CPU. For instance, we apply the controller and left click to generate an input signal in the controller and transfer it to the CPU through the bus. The CPU recognizes the input, does the operation and passes the control signal to corresponding peripherals.

And after the control signal dealt with in the controllers, the peripherals generate output, such as showing figures on the screen or generate sound effects from the speaker.

In the VGA module, we plan to use the frame buffer to display different sprites. The reason we choose to use the sprite graphic display is that we have many pictures such attackers and towers, and their behavior is mutually independent. So we can better control them by laying them on different sprites. So there will be one sprite control part. Meanwhile, we will also design a normal VGA control module, with which we can show every pixel on the screen as in the lab3.

For the controller, we need to do the similar job as we did in lab2. The difference is that we change the peripheral from keyboard to controller. The header file is given online, so we need to write the C file of the controller according to the lecture and some other materials we have researched online.

Next is the storage module, after the initial estimation, we decide to use the 512KB SRAM on the board to store the pictures and short sound effect waveform. To read from and write into the SRAM, we need to design a SRAM controller, which is the key point in this module.

Finally comes to the audio module. In this project, we just need some short music as the sound effects in some specific cases such as attacking the attackers, the attackers die, winning or losing the game at the end and etc. We can refer to the FM Sound Synthesizer part of lab3 and do some revision according to the practical situation.

Algorithm:

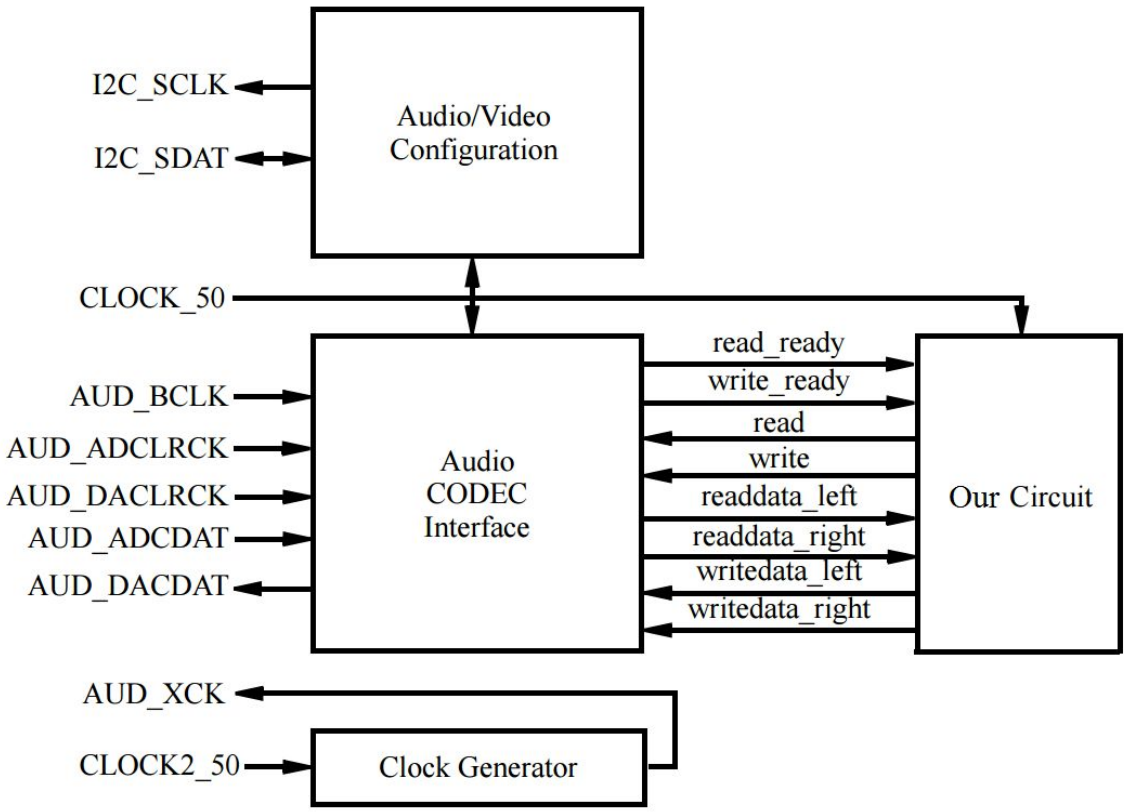
In this section, the basic game logic will be discussed. It can also be considered as a tutorial for this tower defence game.

The screen will be sliced into a grid. A path will be generated based on our setting. A certain amount of pests will occur at the beginning of the path. Then they will move along the path and try to reach the end of the path, the barn that storing all the crops. From the point of the code, each the pest will be considered as a set of coordinates which can represent their current location and a value that represent their health point. The coordinates increment following the pattern of the path. Once the pests are attacked by the plants, their health points will be reduced according to the attack of the plants. If their health points become zero, pests die which means that their coordinates will be eliminated from the program. If their coordinates turn out to be identical with the destination, the coordinates of the barn, they will vanish too and the value displayed on the barn will accumulate. Once this value reaches a certain number, the player fails. On the other side, the player can use XBOX controller to build the defence in order to defence the barn. Typically, the player can press the up, down, right, left buttons to move a cursor through the grid, which will highlight the corresponding blocks in the grid.

When the player press one of the control buttons, then the cursor will become a plant. And the play has to choose which plant to be planted in the ground. There are three different kinds of plants, one can do the normal attack, one can slower down the pests, and one can do damage over an area. All these plants will have certain attack ranges. The program will always calculate the distance between the plants and the pests. Once the distance is smaller than the attack range, the program can calculate the trajectory and the bullet will be generated along the trajectory periodically. The plants will always attack the first pests that move into their attack range. Their target will change until the first pests vanish or they just move out of the attack range. Once a pest is eliminated, the player will gain some coins as a reward. These coins can be used to plant new plants or upgrade original plants. As a wave of the attack is dissolved, the player will have several seconds to build or upgrade. Then the next wave of more pests will be ready to try to reach the barn.

Hardware:

Audio Controller Implementation



The left-hand side of the above figure shows the inputs and outputs of the system. These I/O ports supply the clock inputs, as well as connect the Audio CODEC and Audio/Video Configuration modules to the corresponding peripheral devices on the board. In the middle of the figure, a set of signals to and from the Audio CODEC Interface module is shown. These signals allow the circuit depicted on the right-hand side to play the background music via speakers.
The system works as follows. Upon reset, all the music data will be stored in the RAMs of on-board peripherals. To output sound through the speakers a similar procedure is followed. Our circuit would observe the write_ready signal, and if asserted write a sample to the Audio CODEC by providing it at the writedata_left and write-data_right inputs and asserting the write signal. This operation stores a sample in a buffer inside of the Audio CODEC Interface, which will then send the music data to the speakers at the right time.

VGA Display Implementation

The VGA screen we use in this project is 640*480. In order to save memory and minimize redundancy, we divide the screen into 300 grids of 32*32 pixels. (640/32 = 20, and 480/32 = 15) Design the monsters, towers, bullets, background textures, icons, and other function buttons with this fix size of 32*32.

The UI will consist of three parts: Information bar, map area, and control panel. The information bar is on the top of the screen showing LIVES, COINS, and LEVEL; the map lies in the middle displaying game area, where player have control to place tower; and on the bottom is the control panel includes START, TOWER SELECTION, and PAUSE.

One of the main challenges in displaying is to make the motion of monsters and bullets smoothly. In order to generate those elements continuously without overlapping display, we implement sprites. Each monster have two sprites for their normal state and attacked state. Towers are displaying using sprite for placing at any designed spots. There are two background textures, grass and road on sprite, manipulate the position of two texture we are able to implement the the monster attacking route. Finally, icons and control buttons are also implemented on sprite.

In a video framebuffer display each pixel's display requires 3*1Bytes RGB values. Summing up, in this game there will be three monsters each with two states (Normal/Injured): 3*2*32*32*3*1Bytes; Four towers: 4*32*32*3*1Bytes; One bullet: 32*32*3*1Bytes; Start and pause buttons: 2*32*32*3*1Bytes; Live and coin icons: 2*32*32*3*1Bytes; And two background texture: 2*32*32*3*1Bytes. The total memory for images is roughly 52.5KBytes.

Elements	Image list	# of images	Size (pixels)	Actual Total Size (KBytes)
Monster	Fly	2	32*32	6
	Worm	2	32*32	6
	Locusts	2	32*32	6
Tower	Corn	1	32*32	3
	Eggplant	1	32*32	3
	Lettuce	1	32*32	3
	Beet	1	32*32	3
Background	Road	1	32*32	3
	Grass	1	32*32	3
Bullet	Bullet	1	32*32	3
Icons	Heart	1	32*32	3
	Coin	1	32*32	3
Button	Start	1	32*32	3
	Pause	1	32*32	3
Total:		**17**		**51**

XBOX Controller Implementation

The controller we choose to work with our hardware is Xbox 360 wired controller. The game receives the operation signals from the controller to move the pointer and build/remove towers. To be specific, D-pad will be used to control the pointer. A button will be used for confirm and build. B button will be used for exit, remove and cancel. The controller is shown in the figure below.

The Xbox 360 controller for windows can be directly connected to the SocKit Board and communicate with the CycloneV Soc FPGA through USB port. The Xbox 360 controller is in brief a USB compound device that consists of a USB gamepad and a two-port USB hub. The USB cable and the built-in memory unit connector carries five signals, four of which are standard USB signals and the rest is a clock signal. The input report of the controller is 20-byte HID report format. Data in Offset+2 contains the information of digital buttons including the D-pad. Bit0 is D-pad up, bit1 is D-pad down, bit2 is D-pad left and bit3 represents D-pad right. Offset+4 is unsigned 8-bit A button signal, and Offset+5 is unsigned 8-bit B button signal.

Thus a game controller driver should be built using SystemVerilog and the interface for the game controller will be implemented in C. The Sockit board provides USB interfaces with SMSC USB3300 controller. A SMSC USB3300 device is used to interface to a single Type AV Micro-USB connector. The device supports ULPI interface to communicate to USB 2.0 controller in HPS. The ULPI PHY can operate in Host or Device modes. In the project, the PHY should be configured to device mode to receive the signals from the game controller.

Milestones:

1	Design and display the structure of game map on the screen Design different models for pests and plants in the game
2	Work on and implement the behaviour of the controller Build the sound effect and background music Achieve code and build the basic level of the game
3	Achieve level increment and upgrading the plants Finish coding software and hardware configuration Testing and debugging the game