Columbia Defense Video Game

Design for CSEE 4840 Project

Team Members:

Chao Li(cl3169)

Zhefeng Xu(zx2152)

Yang Bai(yb2310)

Tianlei Zhou(tz2212)

March, 2014

1. Project Introduction :

In this project, we plan to design and implement a tower defense game called “Columbia Defense”, which is based on Columbia University campus. The user can make the strategy, set up various defensers in the pathway in order to kill the enemies and protect our Columbia campus. The attackers include people from other Ivy league universities and the defenders are undergraduate, MS, PHD, TAs, Professors of Columbia University. All of the defenders have different kinds of features and attack techniques and furthermore in the bonus system, killing the attackers will make some money for the user, which can be used to upgrade our defender. The upgraded defenders have more powerful weapons and abilities. In the screen, we have 5 parellel pathways towards our campus and we should build our defenders on each path to accomplish the goal of protection. Once one of them is occupied by the attackers, our game will be over. The use can choose to play again and exit. As the user pass the elementary level of our game, next higher level will be available, where there are more powerful attackers.

There are some key points of this project with which we need to handle carefully. First is the VGA display, because there are lots of different attackers, defenders, special effects and campus background. We need to rewrite the driver and use different methods to solve the static and dynamic figures, such as sprite. Second key point is the algorithm issue. For both the attackers and the defenders, we need to set them walking along the designed pathways, the different behavior and movement when they attack or are attacked. For the defense facilities, what’s more, we need to design different characteristics according to upgrading. And how many times that our defenders can destory different attackers.

2. Implement details

<A>. VGA Display:

In our project , we are going to show the following elements:

1. Background including the map of the game, the location we set our defender, the path towards our campus, etc.
2. Six different defenders and four different attackers and each has different appearance.
3. Different attacking, hitting and moving effect for attackers and defenders.
4. The game money system and defenders selection part should be shown on the specific part of the screen.

To realize the background and the appearance of the attackers and defenders, we will store these elements as fixed figures in the SRAM on the borad. And for realizing the attacking action and the effect of being hitted dynamically, we plan to prepare 3 to 4 figures for a certian action and display all the dynamic actions by refreshing the sprite frames frequently.

Consider that the total screen is 640x480 pixels, and we will set 5 paths towards the campus and each paths will have 9 squares for setting up defenders and for attackers going through. So that it is reasonable to set each squares as 60*80 pixels and 55*90 pixels for defenders and attackers for a bit sense of 3-Dimension. In the whole game, the game information such as the amount of money occupies a sprite. We assume that the effects of defender being hitted are the same so we only need each sprite for the generation of each defender. And it is more complicated for attackers because we need one more sprite for different motions after they being hitted(such as be killed or stuned to stop moving). We will set the attacking effect as the top level and then the attackers, following the defenders and the last to be the background and the game information.

<B>. Audio Controller:

1. Background Audition:

Since this time the capacity of the SRAM is fairly enough for us, so it is reasonable for us store the music files in he SRAM in the Sockit board rather than store in SD-card. Thus this time the SD-card drive program file can be well omitted.

2. Game Sound Effect:

we would look for some proper short audio files, once there comes a request(namely, some behavior of the heros in the game), the corresponding short audio will be triggerred and play for one time.

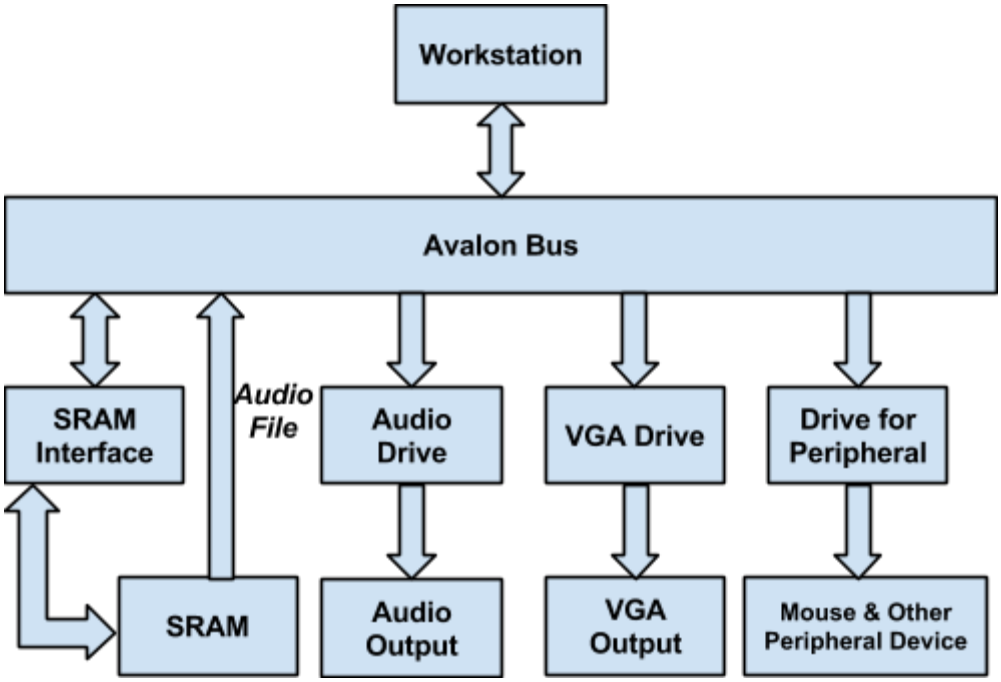
For this case, we have to consider that once there come two requests at the same time(a new requet comes before the ongoing one finishes), we would have to play two audio files concurrently. It is just a similar case if more than 3 requests occur concurrently. Therefore, mutilple-track player will be required.

<C>. External Equipments:

For this project, we will mainly use the mouse as our first choice to do the basic operation, such as selecting different kinds of defenders.

As we designed above, we have 5 paths with 45 set up locations in total and there are six different kinds of defenders. Furthermore, we may have some other functional buttons such as upgrade, pause. So there will be appoximately 55 tiles in total on the paths. When moving the mouse, there will be a cute pointer icon moving correspondingly with the mouse. And according to the position of the tip pixel of the pointer, we can decide which tile the mouse is cover on. In that case, once the CPU recognizes the input signal of a left click, the system will realize its corresponding function.

3. Block Diagram & Interface Implementation:

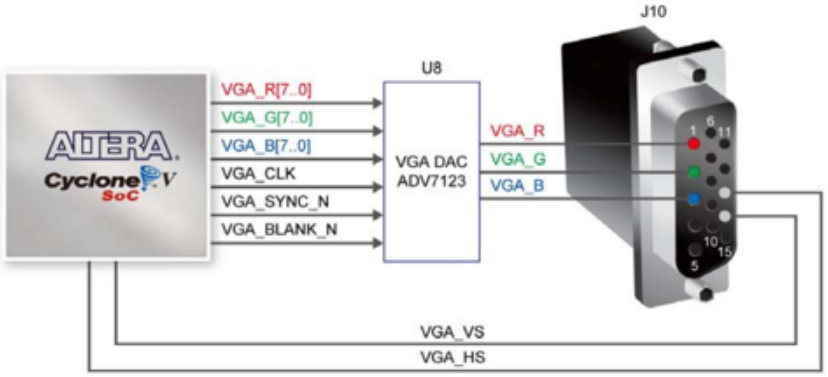


INTERFACES:

As is shown in the above diagram, hardware-to-hardware blocks is rare in our system design, that is, almost all the interconnect between blocks are via ports with corresponding commincation protocals. Thus, we are supposed to pay more attention on protcal design rather than puring low-level device timing design.

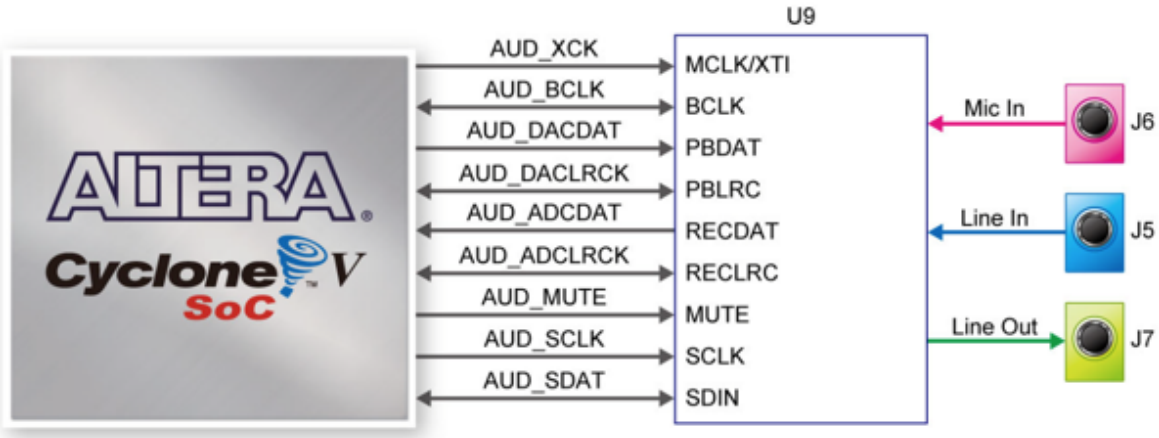
<A>.VGA output:

The board includes a 15-pin D-SUB connector for VGA output. The VGA synchronization signals are provided directly from the Cyclone V SoC FPGA, and the Analog Devices ADV7123 triple 10-bit high-speed video DAC (only the higher 8-bits are used) is used to produce the analog data signals (red, green, and blue). It could support the SXGA standard (1280*1024) with a bandwidth of 100MHz. The following figure gives the associated schematic.



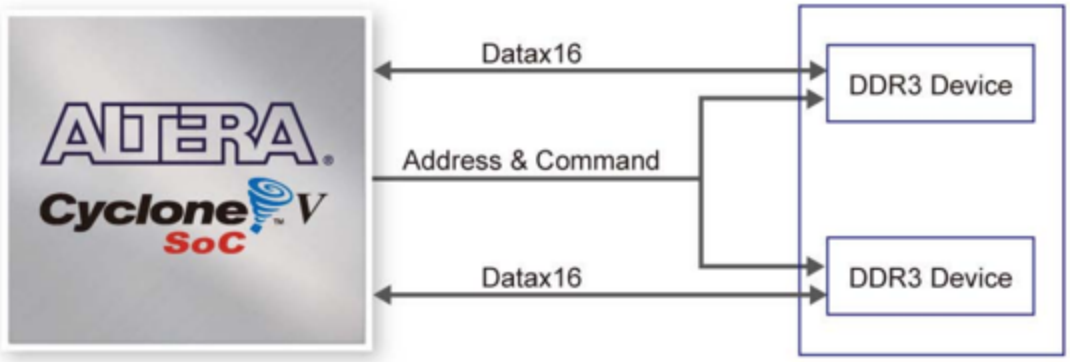
<B>.Audio output:

The board provides high-quality 24-bit audio via the Analog Devices SSM2603 audio CODEC (Encoder/Decoder). This chip supports microphone-in, line-in, and line-out ports, with a sample rate adjustable from 8 kHz to 96 kHz. The SSM2603 is controlled via a serial I2C bus interface, which is connected to pins on the Cyclone V SoC FPGA. A schematic diagram of the audio circuitry is shown in the following figure.



<C>.SRAM interface:

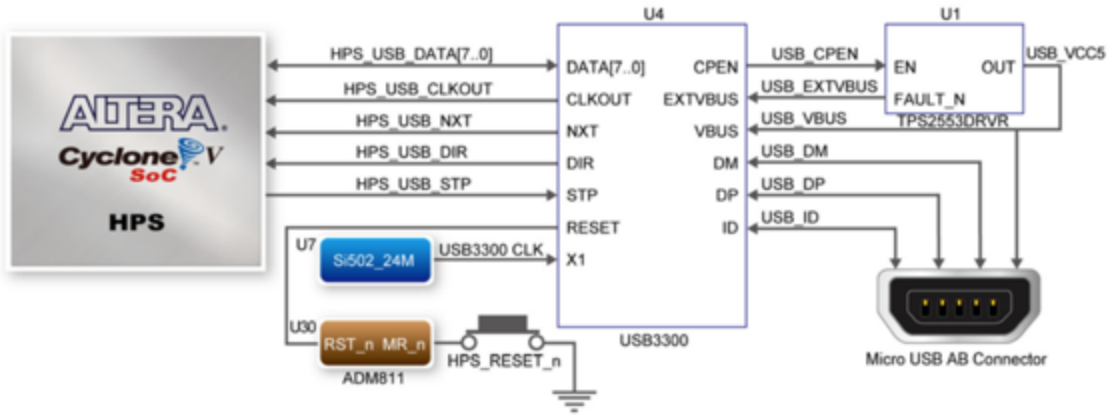
The board supports 1GB of DDR3 SDRAM comprising of two x16 bit DDR3 devices on FPGA side. The DDR3 devices shipped with this board are running at 400MHz if the hard external memory interface is enabled, and at 300MHz if the hard external memory interface if not enabled. The following figure shows the connections between the DDR3 and Cyclone V SoC FPGA.



<D>.Peripheral Device:

The board provides USB interfaces using the SMSC USB3300 controller. A SMSC USB3300 device in a 32-pin QFN package device is used to interface to a single Type AB Micro-USB connector. This device supports UTMI+ Low Pin Interface (ULPI) to communicate to USB 2.0 controller in HPS. As defined by OTG mode, the PHY can operate in Host or Device modes. When

operating in Host mode, the interface will supply the power to the device through the Micro-USB interface. The following figure shows the schematic diagram of the USB circuitry.



4. Algorithm:

Here we want to briefly introduce our idea about the implementation algorithm of the behavior control of every attacker and defender, including attack behavior, attacked behavior and upgrade behavior.

As for the attackers, they have two behaviors: move, attack and die. When the attackers appear at the beginning of the path, they will follow the designed path at the initial speed and move forward normally. When they are blocked by our defenders, they begin to attack and at last clear the defenders and all the attackers have the same behavior. However, when they are killed by our defenders, it will die and disappear gradually, which can be implemented by setting a counter for an attacker and Every time the attackers are attacked again, refresh the counter and zero means the attacker disappearing. The above is similar for the defenders.

For the defenders, there are two behaviors: attack, upgrading and die. Every defender has its own attack range, such as 100 pixels, 200 pixels. Once the attacker is built, the boundary is fixed. So if there are any attackers entering this boundary, the defender will attack. Similar with drawing a line, we can calculate the relative position according to the x and y coordinates of the attacker and the defender. But here we plan to use special effect of small circle instead of the line. The defender will follow the first attacker coming into its attack range until the attacker gets out of the boundary or dies. Then the tower will turn to attack the second one. This behavior can be implemented by numbering every attacker in the range from 1 to n, and once the first attacker goes far away or dies, every number decrease by 1, and the tower always attacks the number 1 attacker. Different defenders have different attack methods, such as decceleration, explosion, ordinary attack.

The other behavior is upgrading. Every tower can be upgraded to achieve stronger power and larger attack range.

5. Milestones:

Milestone 1 : (Apr 1)
Design and display the structure of game map on the screen: pixel positions and graphic design.
Design different models in the game: different university guards and attackers.

Milestone 2: (Apr 15)
Work on and implement the character behavior with mouse and hardware configuration.
Build the sound effect, the bonus and upgrading system.
Achieve code and build the basic level of game.

Milestone 3: (Apr 29)
Achieve changing and building different levels.
Finish coding software and hardware configuration.
Testing and debugging the game.