BOMBERMAN

CSEE 4840 Embedded System Design

CH2788 Cheng-Han Hsieh

JW 2862 Jian Wang

MW2724 Ming Wang

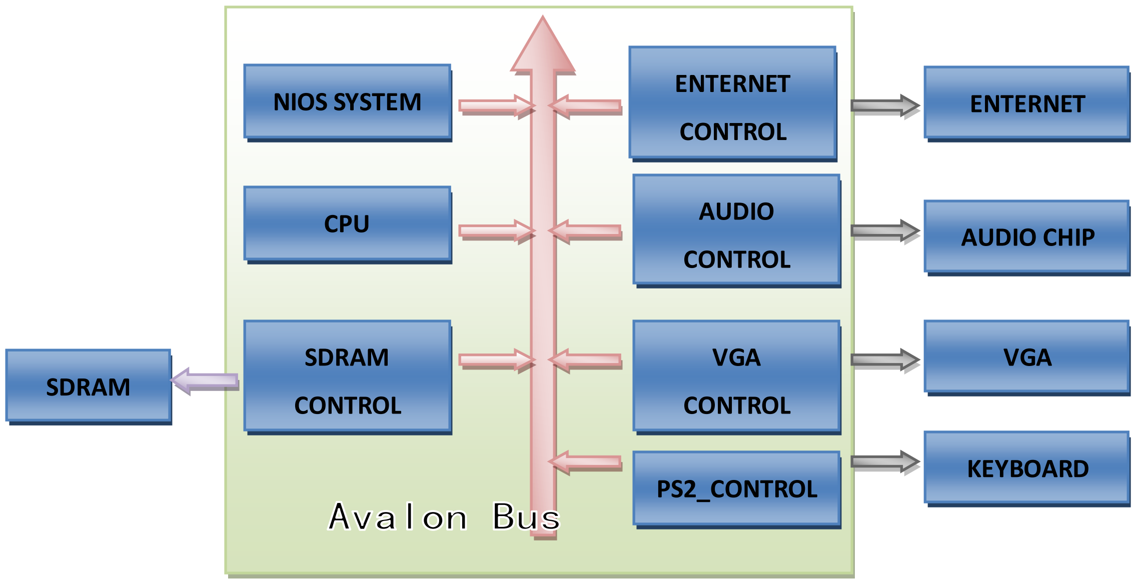
SC3198 Shang-Ming Cherng

1. Project Instruction

In this project, we intend to implement a popular game called BOMBERMAN. It will be a game of multiple players. Each player plays the game in separate machine and communicates with others through network. The general goal for game player is to kill each other by strategically placing bombs which will explode after predefined period. Bombs in the game have predefined effective hurt range, within which any player will be killed. After several games, we will decide the winner who kills the most people.

2. Design Block

The following is the design block of our project. It is composed of CPU, SDRAM CONTROL, SDRAM CONTROL, ENTERNET CONTROL, AUDIO CONTROL, VGA CONTROL and PS2 CONROL. We will describe how they will be designed in this report.



3. Software/ Hardware Implementation

In the part we will describe how will we implement our project in software and hardware. Software and hardware have identical control table that is used to control character and rendesr the scene on screen.

a. Software

Software is mainly responsible for characters controlling, event processing and network communication.

Players will use arrow keys on keyboard to make Bomber men move. For each movement software will check whether the Bomber men can do that or not through checking control table. After checking we can decide how to change the position of characters.

There are several events we have to address in the game. For example, we need to build timer for bombs' explosion. Each player will have their own timer that should be synchronized. By using the timer we can know when and what bombs will explode. In addition, we also need to detect the death of players. Once a player is died, we will remove the player from screen and check game is over or not.

Finally, we need to implement the network communication part. It checks the synchronization and transfers data. If some players are not synchronized, we make stop for a little while to do synchronization.

b. Hardware

Hardware is responsible for sprites' rendering and audio synthesize. We will store lots of sprite arrays in the hardware and render these patterns on screen by control array. We plan to make each sprite size as 32x32 pixels and the whole screen is 640x 480. So the control array should at least 20x15. However, considering the development of our project, we make the control array as 120x15 that can render at least 64 different kinds of tile in one position. For character rendering, we will make its coordination as 640x480. It means Bomber Man will not walk one tile by one tile, but one pixel by one pixel.

Regarding the audio generation, we plan to generate some sounds effects, like explosion sounds, game starting music, and game ending music, etc. Each sound will be trigger and terminate by software. We will describe more details in the following parts.

4. VGA CONTROL

a. Game Graphics

In the lab3, we have learned to implement a video display. We were able to store our character sets and graphics in the RAM.

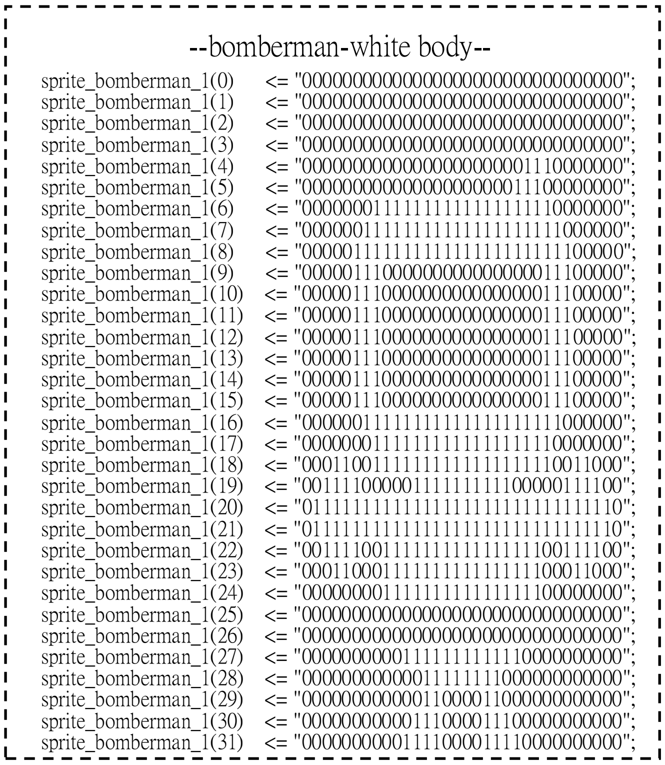
We will generate out graphics in an array representing 32x32 pixels, each coded in hexadecimal. Each 32x32 pixels array on the screen will be represented by two strings of nine hexadecimal values. Each group of nine values will represent a mapping in one color. The first term in the string will represent the value for the color of the following pixel map. We will then proceed to read the second string of values to overlap the same 32x32 pixels space on our screen. The second string will represent the second color, with no pixels overlapping the first layer. This approach allows us to create 32x32 pixels array with two colors. For convenience, the game engine will read 16 values at a time.

b. Graphics example

We decide to make the Bomber Man as a 32x32 pixel character. The most important thing is that Bomber Man will walk on four different directions- right, left, up, and down. So sprites should be different for Bomber Man’s movement.

We use 5 colors to build a Bomber Man, thus we use 5 patterns to form a character. Each of the patterns has different color.

The below is one pattern of the Bomber Man:



```
--bomberman-white body--
sprite_bomberman_1(0)    <= "00000000000000000000000000000000";
sprite_bomberman_1(1)    <= "00000000000000000000000000000000";
sprite_bomberman_1(2)    <= "00000000000000000000000000000000";
sprite_bomberman_1(3)    <= "00000000000000000000000000000000";
sprite_bomberman_1(4)    <= "00000000000000000000001110000000";
sprite_bomberman_1(5)    <= "00000000000000000000011100000000";
sprite_bomberman_1(6)    <= "00000001111111111111111110000000";
sprite_bomberman_1(7)    <= "00000011111111111111111110000000";
sprite_bomberman_1(8)    <= "00000111111111111111111111000000";
sprite_bomberman_1(9)    <= "00000111000000000000000011100000";
sprite_bomberman_1(10)   <= "00000111000000000000000011100000";
sprite_bomberman_1(11)   <= "00000111000000000000000011100000";
sprite_bomberman_1(12)   <= "00000111000000000000000011100000";
sprite_bomberman_1(13)   <= "00000111000000000000000011100000";
sprite_bomberman_1(14)   <= "00000111000000000000000011100000";
sprite_bomberman_1(15)   <= "00000111000000000000000011100000";
sprite_bomberman_1(16)   <= "00000011111111111111111110000000";
sprite_bomberman_1(17)   <= "00000001111111111111111100000000";
sprite_bomberman_1(18)   <= "00011001111111111111111110011000";
sprite_bomberman_1(19)   <= "00111100000111111111100000111100";
sprite_bomberman_1(20)   <= "01111111111111111111111111111110";
sprite_bomberman_1(21)   <= "01111111111111111111111111111110";
sprite_bomberman_1(22)   <= "00111100111111111111111100111100";
sprite_bomberman_1(23)   <= "00011000111111111111111100011000";
sprite_bomberman_1(24)   <= "00000000111111111111111000000000";
sprite_bomberman_1(25)   <= "00000000000000000000000000000000";
sprite_bomberman_1(26)   <= "00000000000000000000000000000000";
sprite_bomberman_1(27)   <= "00000000011111111111110000000000";
sprite_bomberman_1(28)   <= "00000000000111111111000000000000";
sprite_bomberman_1(29)   <= "00000000000011000011000000000000";
sprite_bomberman_1(30)   <= "00000000000111000011100000000000";
sprite_bomberman_1(31)   <= "00000000001111000011110000000000";
```

5. AUDIO CONTROL

The audio part is similar with the FM sound synthesizer of the lab3,The basic FM equation is

$$x(t) = \sin(\omega c\, t + I \sin(\omega m t))$$

Where x(t) is the amplitude at time t. ωc , is the carrier frequency(the fundamental tone we hear), ωm is the modulating frequency, and I is the modulation depth. The timbre of the sound is largely determined by the ratio ωc / ωm , which is generally set to an integer ratio .The fundamental frequency of musical notes follow an exponential scale. The A above middle C is 440 Hz, and going up an octave doubles the frequency. We make the music in the VHDL part, create four different music for the software to use in the video game. Just change the modulating frequency and the modulation depth to make the musical notes we wanted. However, how long the time the musical note will last? We use a counter to control the time the notes will last. For example, the clock is 50MH, we make the counter count to be 50M, the note will last one second.

About the explosion sound, I want to make a whole table which is derived from some .wav sound file, when we need to output the explosion sound, write some data to the component and scan the table for once which may last 1 second.

6. ENTERNET CONTROL

One essential part of this project is networking and synchronization between different terminals, i.e., players. (4 players maximum) All terminals will be interacting with each other through Ethernet connection. As in Lab2, Altera DE2 board's DM9000A chip that contains a fast 10/100 Mbps transceiver is employed to communication under CSMA/CD protocol.

During the communication, 4 players will send UDP/IP packages to each other and there is no concept like master or slave, in other words, all terminals are treated equally. Mainly two kinds of packages are passed around: displacement event and bomber event. Whenever one terminal makes a movement or places a bomber by pressing the keyboard, corresponding package will be send to other 3 terminals. Once receiving the package, terminals would update their own screens after package validation. On the other side, the sending terminal would update its own screen on the successful transmission of package.

The explosion of bomber is controlled by each terminal's clock. All clocks will be started at the same time. Each will counts automatically afterwards. A bomber explosion schedule table is maintained in each terminal for the purpose of next explosion event scheduling. On the receiving of bomber event package, the bomber explosion schedule table will be updated in receiving terminal. (Intuitively, sending terminal will update its own table) According to this table, each terminal can handle the explosion and render the screen independently and simultaneously. For example, if one entry of terminal A's table indicates explosion of bomber "k" at "t" seconds, terminal A will handle the explosion of bomber "k" once its clock counts at "t" seconds.

7. CONCLUSION

In our project, we store most of data in the DE2 board. For addressing each event, like bomb explosion, we can dynamically declare memory and store it in an event queue. After event over we free it and remove it from queue. So the part of software will not occupy lots of memory. For a 512K SRAM it should work. However, it may be involved of synchronization problems. We need to design a robust mechanism that can make sure players can always play synchronously.

8. MILESTONES

Here we describe what we expect to finish in each milestone.

a. Milestone I

- Construction of game map and basic sprite designing.
- Character sketch and movement controlling.
- Events processing- bomb explosion.
- The synthesis of sound of bomb explosion.

b. Milestone II

- Allow two players to play synchronously.
- Start and end frame design.
- Events processing- death of players and detection of game over.
- Music synthesis of game start, end, etc.
- 3D like sprite development.

c. Milestone III

- Allow four players to play synchronously.
- Record players' scores and show the rank list on screen.
- The synthesis of sounds of background.
- Small animation for Bomb man's walk.
- Fancy feature development- players can throw bomb, time bomb, etc.