# Fantastic Tetris

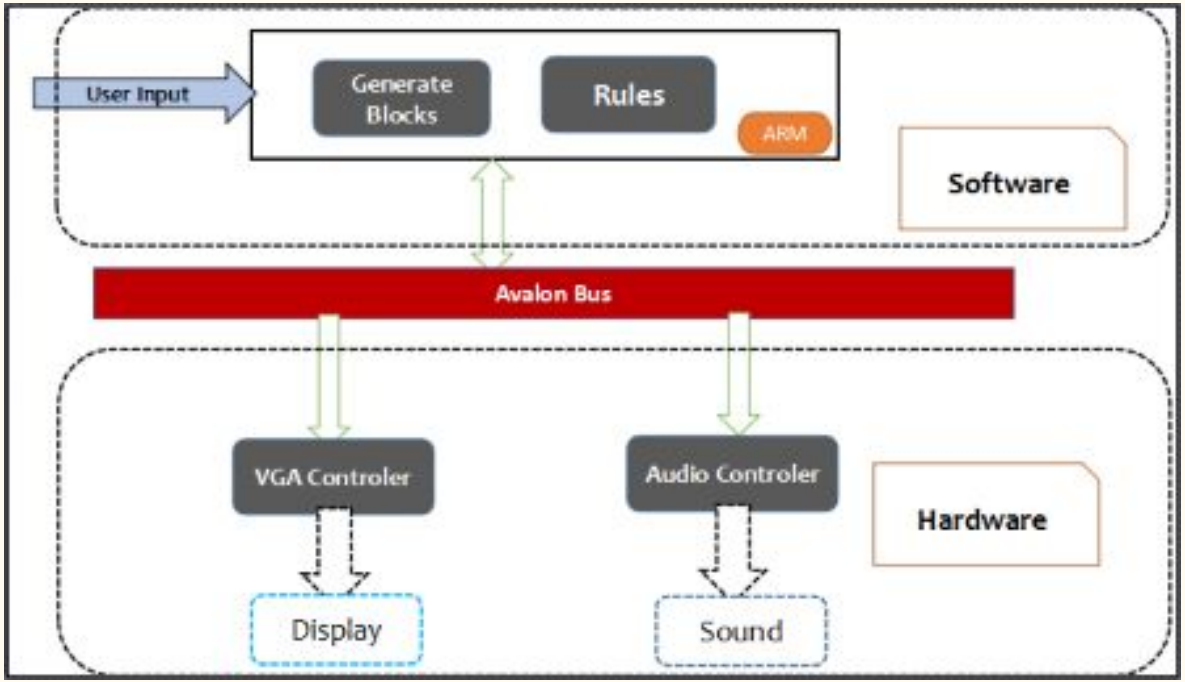
By Weipeng Dang; Benjie Tong; Yanbo Zou; Yiran Tao

# Introduction:

The fantastic Teristic is a variation of a normal Teristic. The game will automatically generate three different blocks into the screen for the users to choose. The user can put the blocks into the grid(10*10 in dimension) in anywhere the user like. After the user put one block, the block is fixed in that position and can’t be moved. After the user put all the three blocks into the grid, the game will check if the game is terminated or not. If it is not terminated, the grid will give three more blocks for the user to continue the game. The user should try to put blocks into the grid in order to form a line. If one horizontal or one vertical line is detected in the grid, that line is cleared.

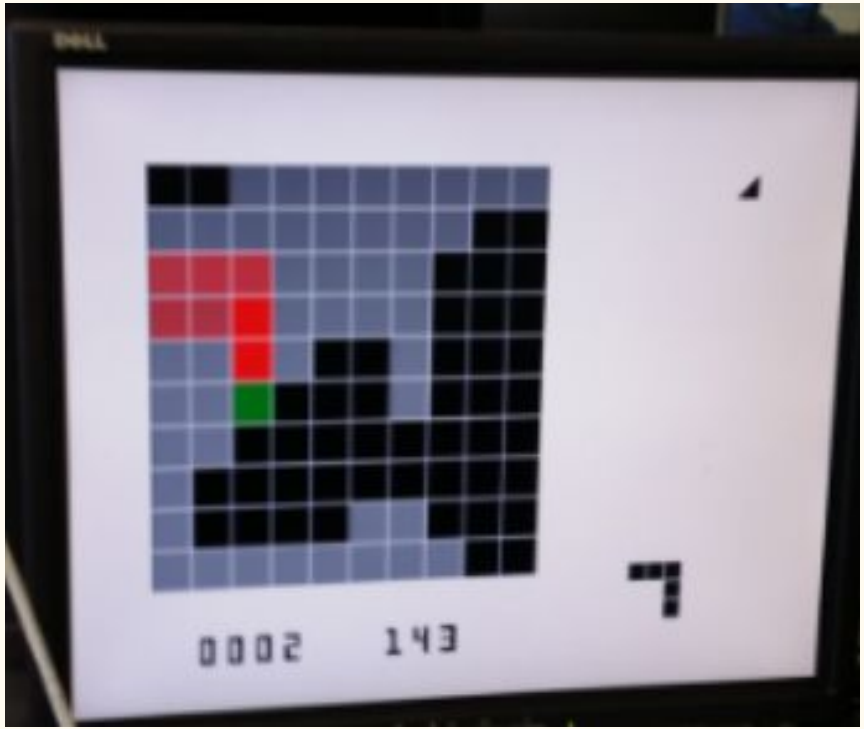
# Hardware and Software Connections:

In software, the C language will be used to implement all the controls of the system. It first generates the three blocks in the screen and then keeps getting the user's inputs to control the screen display by using the VGA controller in the hardware part.



# User Interface

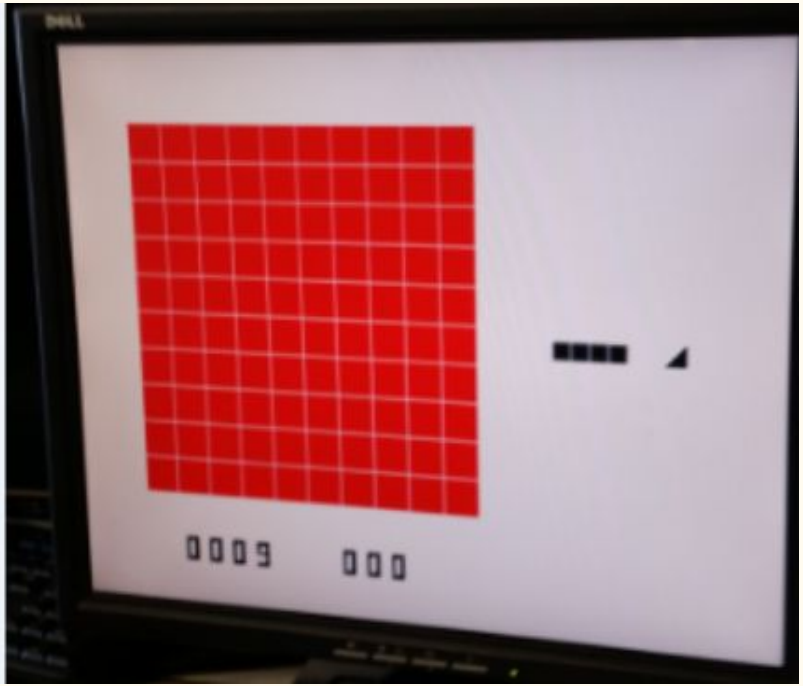
As showed in the graph, the grid is consist of 10*10 blocks. The “red” blocks are the blocks are put in the grid and can’t be moved(except one line formed and the red line is cleared). The “gray” block means that that block has not been put but can’t be put in the position because another block has already been put in the location. The “green” block has not been put and can be put in the location. The three blocks in the right of the grid are used for the user to select. After the user select and put the block into the screen, the block will disappear. If one line is cleared, the score will be recorded at the bottom of the screen.



# End Of Game

There are two data shown in the bottom of the screen. One in the left(4 bits) is to record the score. If the user successfully clear one vertical line or a horizontal line, there will be one score stored.

The three bits data in the right bottom screen is the time left. In the beginning the time is 100 seconds and it will be decreased. After 100 seconds passed, the game will terminated automatically, And the grid becomes all green, then all grey and then all red if time is zero. Also, as a bonus, if one line is cleared, we give ten seconds increment.



# End of Game

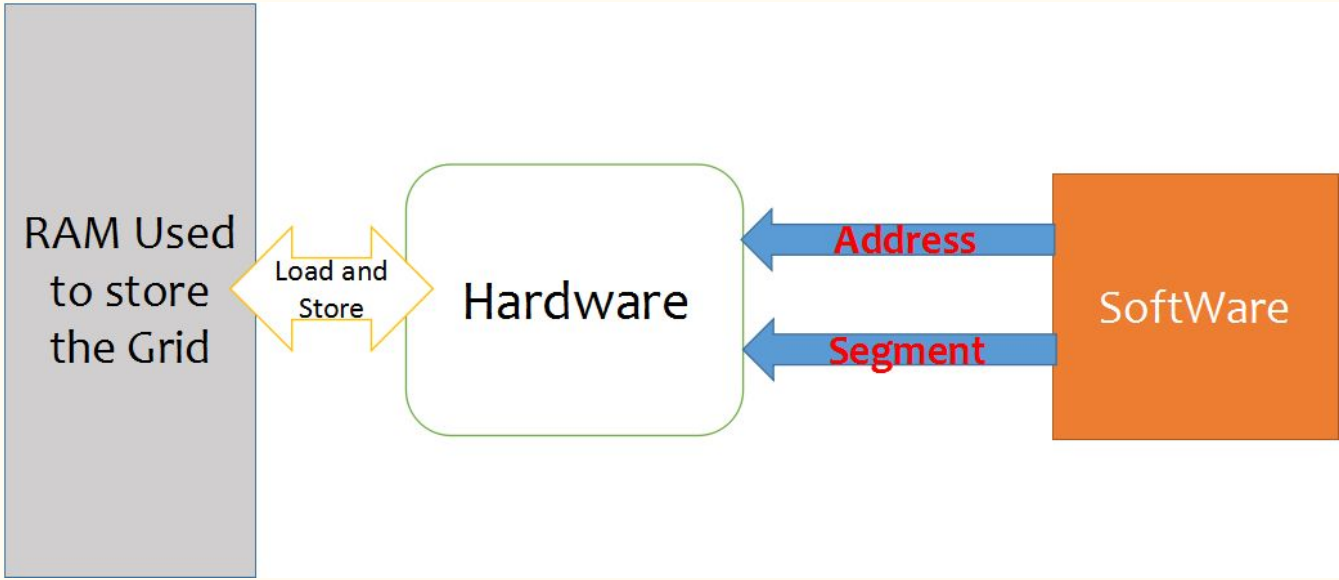
The design of the end is to use the thread method to keep record the time. The “start” is the time the game is started and we keep track the time by calculating the “dif”. The time will be separated into three parts and passed into hardware by using three “write_segment” operations.

```
void *timer_thread_f(){
    start = time(NULL);
    int TOTALTIME = 100;
    while(1){
        time_t dif = time(NULL) - start;
        write_segments(134, ((TOTALTIME-dif)/100)%10);
        write_segments(135, ((TOTALTIME-dif)/10)%10);
        write_segments(136, (TOTALTIME-dif)%10);
        if (TOTALTIME - dif <= 0 ){
            GAMEOVER = 1;
            }
        }
    }
```

# Communication

The “address” is the grid’s index and the “segment” is the data in the grid.

The hardware stores all the information of a grid by using a RAM module. It loads the gird if it needs and can change and stores the data in the RAM whenever the software indicates a change.

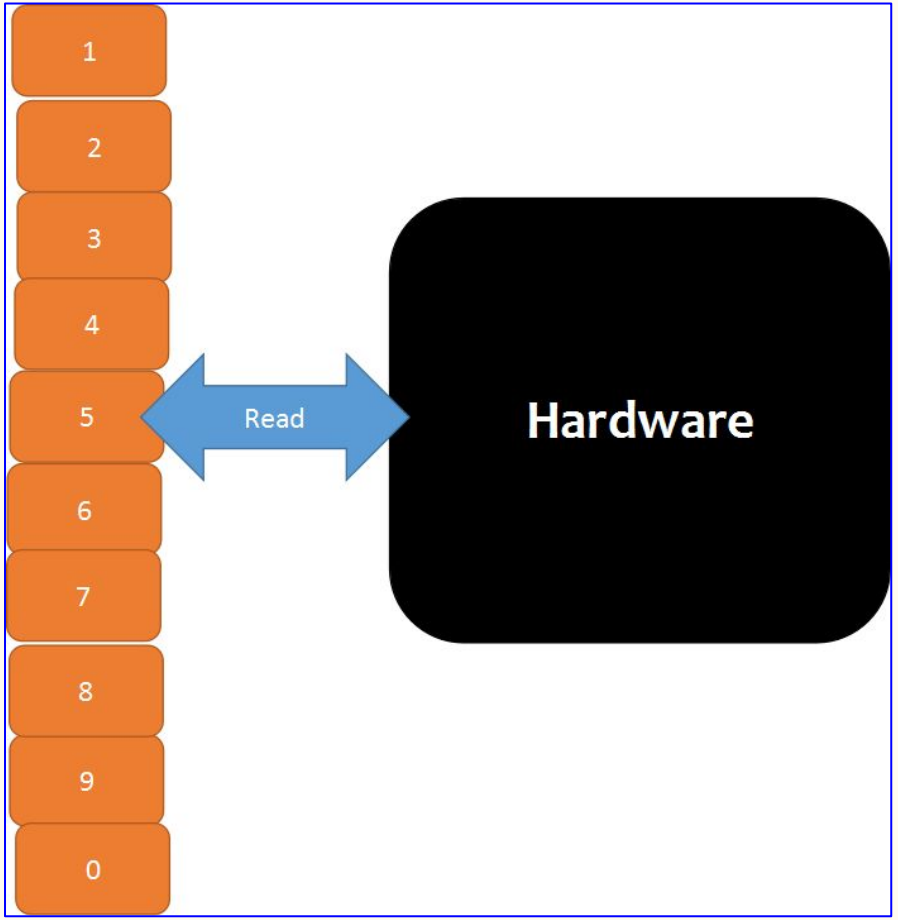


# ROM to Integers

The ten integers are stored in the ROM and can be accessed by the hardware to displayed into the screen.

We use “vcount” and “hcount” to indicate the specific location where the integer showed.

The size of every 23*32 sprites.



# Hardware Design Overview

Most Important File:

VGA_LED: top module for the communication with Avalon MM.

VGA_LED_Emulator: module for all vision accomplishment .

Under VGA_LED_Emulator: One 2-Port-RAM for array data write and read

Ten 1-Port-ROMs for score sprites read

# Hardware VGA_LED

VGA_LED: Communicate with Avalon MM

Basic logic:1. Receive two 8 bit buses, Address and Writedata

2. If Address is less than 128(Address[7]=0, only use is 0-99), pass Address and Writedata to VGA_LED_Emulator and write it in RAM for further use; if address is Address is more than 128 (Address[7]=1, only 10 are used), pass Writedata directly to the Register according to the Address.

# Hardware VGA_LED_Emulator

VGA_LED_Emulator: Achieve all Vision Acomplishment

1. Control the three parts in the screen: Main Grid, Tetris selection, Score Display

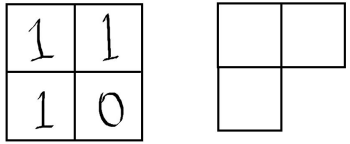
Main Grid: Keep refreshing by the software, pass the data to RAM and according to the hcount and vcount to decide the read address.

Tetris Selction: 14 different type of tetrises embedded in the hardware by directly limit the range of vcount and hcount.

Score Display: Use ROM to store the sprites and read them according to the position.

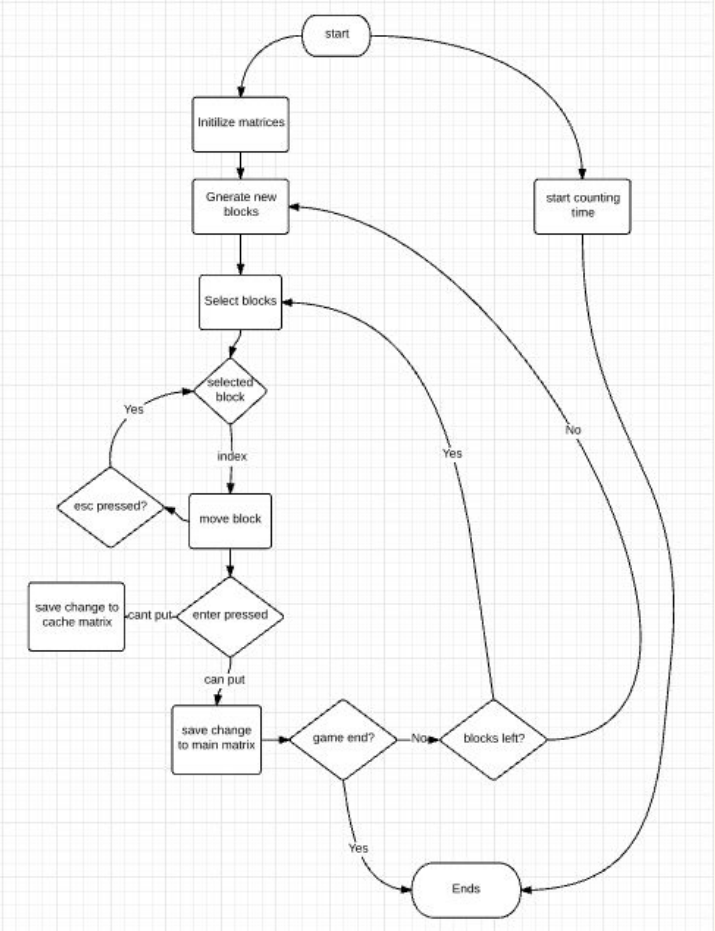
# Software Design

In software, shapes are stored as two dimensional arrays




Use matrices to store game status

- Main matrix for confirmed change
  - 1 - occupied
  - 0 - unoccupied
- Cache matrix for unconfirmed change
  - 3 - occupied and covered
  - 2 - unoccupied and covered
- Timing limit



# Audio And Sound

We use XBOX handle to control the game. Instead of using others' driver, we design our own driver.

# In the future

We are trying to make the game can be played by multiple players. However, if we connect two usb inputs to the game, both of the two players are disenabled.

We are trying to figure out this problem and make the game work in two-player modes.