综合项目（2048游戏）实验报告

——电信提高2101班杨筠松U202115980

1. 实验背景

2048游戏是一款益智类游戏，玩家需要通过合并相同数字的方块，不断合成更高的数字，最终达到2048这个目标。该游戏具有简单的规则和直观的操作方式，因此在移动设备和计算机上广受欢迎。

在微机原理实验中，我们通过2048游戏算法，旨在深入理解微机原理的相关概念和技术，并将其应用于具体项目的实现中。该实验旨在提升学生对计算机硬件和软件的综合理解能力，培养其分析问题、编程实现的能力。

1. 实验目标
2. 理解VGA显示原理并使用
3. 增强对软件和硬件接口设计的认识和理解
4. 使用合适的数据结构手段完成代码实现
5. 实验环境
6. Windows 11操作系统
7. Vivado 2018.3及SDK
8. Image2LCD
9. VGA转VGA线
10. VGA显示屏
11. 实验原理介绍
12. VGA原理介绍

VGA接口的基本原理是通过模拟信号来传输图像数据。图像数据由水平同步信号（HSYNC）和垂直同步信号（VSYNC）进行同步，同时使用红色（R）、绿色（G）和蓝色（B）三个颜色信号来表示像素的颜色。VGA接口的标准分辨率为640x480像素，每个像素可以表示16种颜色。通过调整信号的频率和时序，可以实现不同分辨率和颜色深度的图像显示。

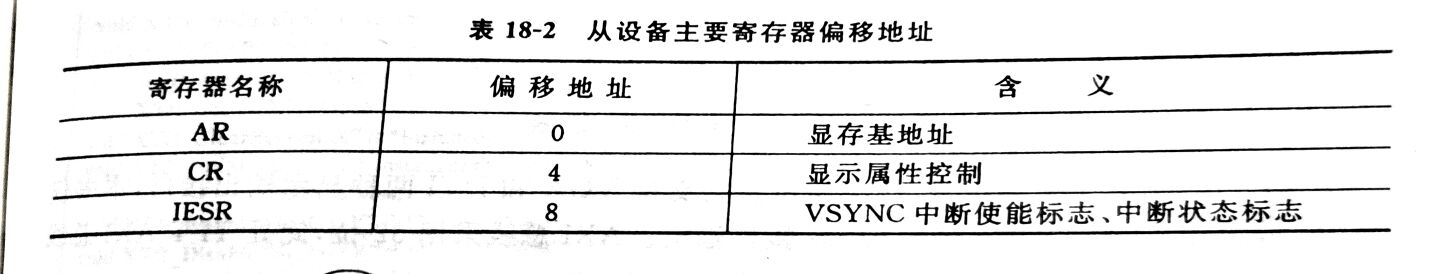
1. 2048硬件平台搭建原理介绍

本次实验需要的硬件平台主要包括：

1. VGA显示控制器TFT
2. MIG 7 Series存储器
3. GPIO核
4. Microblaze微处理器

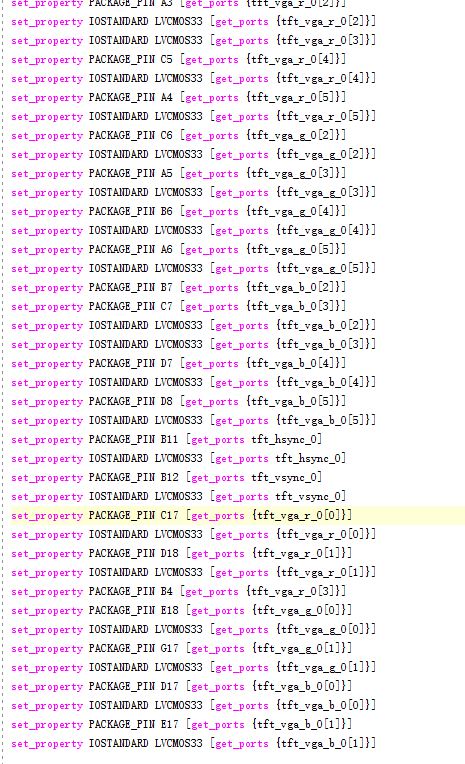
其中重点介绍一下VGA显示控制器TFT的配置，其余按照之前课程连接即可。

VGA显示控制器TFT IP核内部寄存器包括AR寄存器（显存高位地址），CR寄存器（显示属性控制），IESR寄存器（中断状态控制），偏移地址如下图所示，这便要求我们需要在软件中填入相应的值到对应地址。

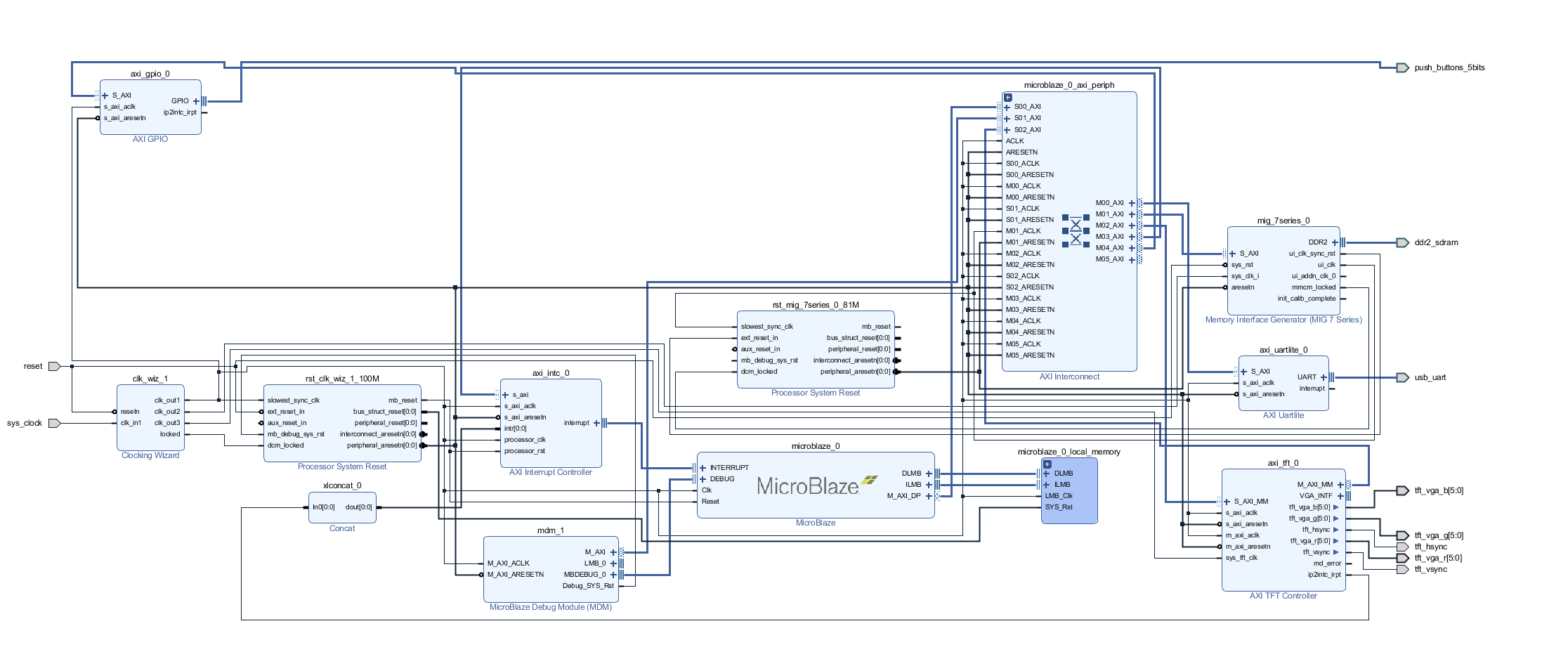


TFT IP核默认的显存容量为2MB，从而AR寄存器仅存储了显存地址高11位，低21位为0，并且默认值为0x00000000;CR寄存器控制显示寄存器的扫描模式和显示、关闭状态，其默认寄存器的值为0x01.

硬件平台搭建过程中，需要将VGA输出信号tft\_vga\_r[5:0]、tft\_vga\_b[5:0]、tft\_vga\_g[5:0]、tft\_hsync、tft\_vsync引出到外部，并且按照如下代码配置约束文件。

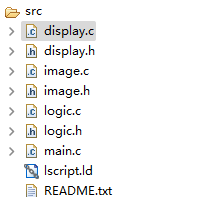


由于需要根据按键状态判断图像转移方向，考虑引入GPIO核来对外部按钮状态进行读取，最终得到完整的diagram如下所示：



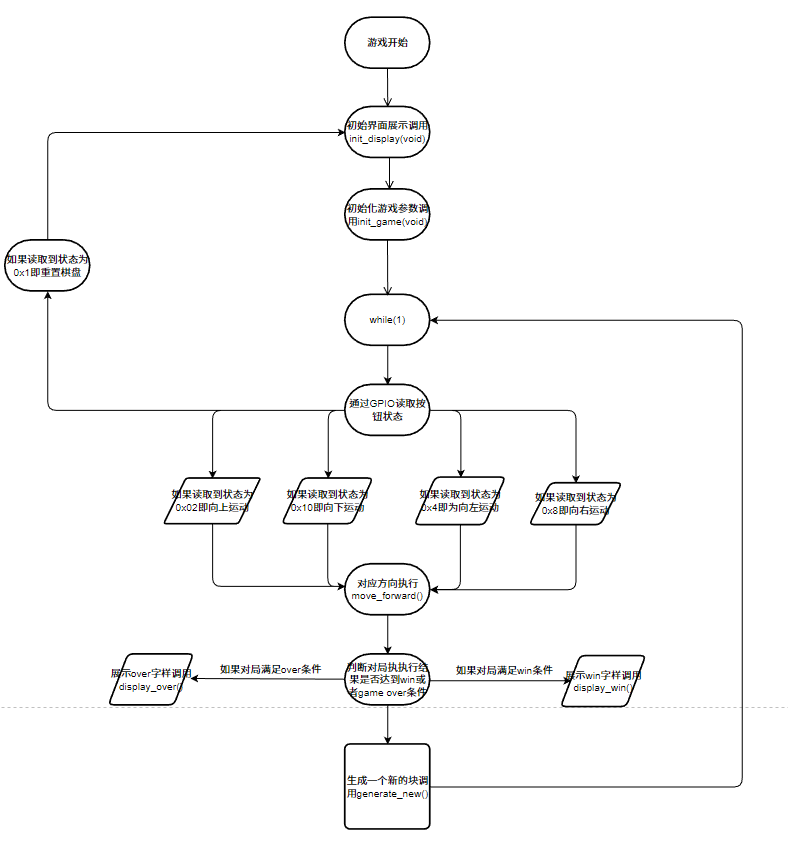
1. 2048软件实现原理介绍

代码采用多文件联编来完成模块化设计，保证了代码之间的独立性，分别设计了main.c、display.c、display.h、image.c、image.h、logic.c、logic.h文件，其中main.c提供主函数入口供处理器调用，display模块主要将需要的图像展示到想要的位置，image模块完成了想要展示图像的存储，logic模块完成了2048游戏主体的代码逻辑，目录结构如下所示：



其中由于内存容量的问题，需要配置对应的脚本文件，具体按照书操作即可。

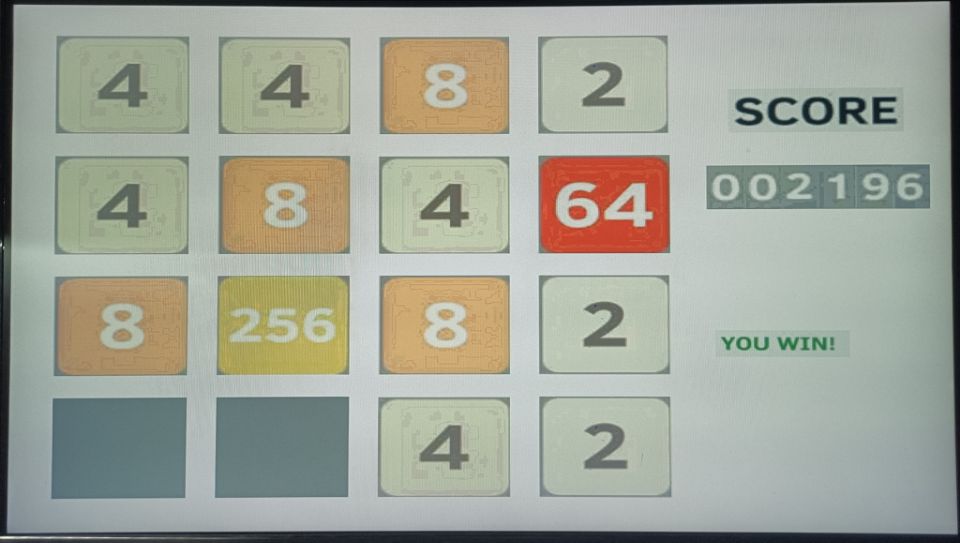
为简化具体代码逻辑而突出代码功能描述，我们可以作出如下的流程图来展现工作流程, 具体代码详见附录：



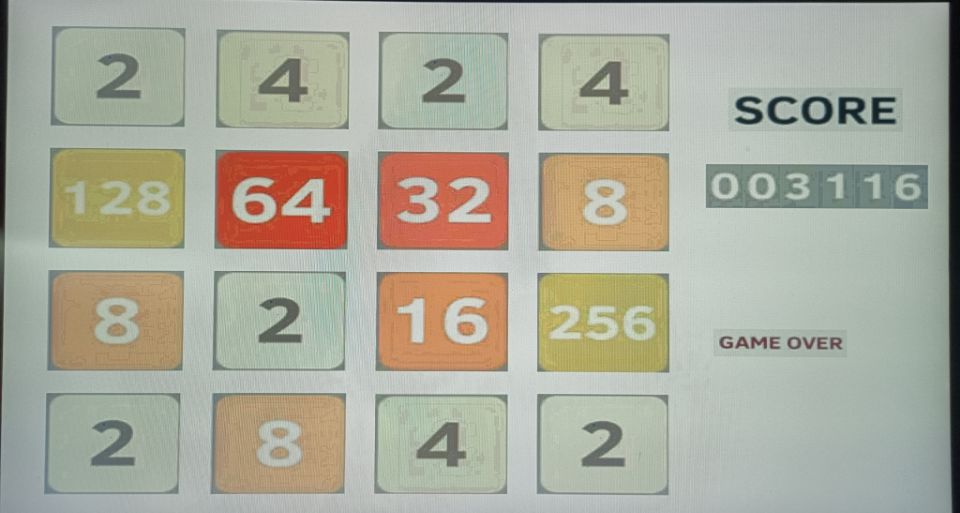
1. 实验结果展示

搭建具体的实验平台后，连接FPGA到VGA显示器，并且连接FPGA到电脑主机上，完成比特流生成后，烧录到FPGA开发板上，运行工程即可。

1. 输出win结果的展示



1. 输出gameover结果的展示



1. 实验总结

本次实验中较为困难的是方块逻辑的设计和硬件平台的搭建，在参考了实验书后，如期完成了实验硬件平台的搭建，对于结果的展示也较为满意，如愿完成了所有的初始设计预期，但是还是因为时间仓促，无法完成滑块动画的展示，是一点点遗憾的。

1. 附录

Display.c代码如下所示：

**#include** "display.h"

**#include** "image.h"

**#include** "xil\_io.h"

**extern** **unsigned** **int** maps[4][4];

**int** **check\_pos**(**int** val){

**int** s = 2;

**for** (**int** i = 1; i < 12; i++){

**if** (s == val) **return** i;

**else** s = s << 1;

}

**return** 0;

}

**void** **init\_display**(**void**) {

Xil\_Out32(TFT\_0\_BASEADDR + TFT\_AR\_OFFSET, TFT\_FRAME\_ADDR0);

**int** i, j;

**for** (i = 0; i < 480; i++)

**for** (j = 0; j < 640; j++)

Xil\_Out32(TFT\_FRAME\_ADDR0 + (4 \* (i \* 1024 + j)), 0xffffff); // clear the screen.

**for** (i = 0; i < 16; i++) display\_bulk(i, 0); // plot each grids for image

**for** (i = 80; i < 120; i++)

**for** (j = 500; j < 618; j++){

u32 color = Xil\_In32((**unsigned** **int**)gImage\_score + ((i - 80) \* 118 + (j - 500)) \* 4);

Xil\_Out32(TFT\_FRAME\_ADDR0 + 4 \* (1024 \* i + j - 10), color);

}

display\_score(0);

}

**void** **display\_bulk**(**int** pos, **int** val) {

**int** pos\_row\_start = 30 + (**int**)(pos / 4) \* 110, pos\_col\_start = 30 + (pos % 4) \* 110;

**int** i, j;

**for** (i = 0; i < 90; i++)

**for** (j = 0; j < 90; j++){

**int** pos\_row = pos\_row\_start + i, pos\_col = pos\_col\_start + j;

**int** addr = (pos\_row \* 1024 + pos\_col) \* 4;

**int** color = 0x808080;

**if** (val != 0) {

**int** color\_addr = (**unsigned** **int**)gImage\_number + 32400 \* (check\_pos(val) - 1) + ((i \* 90 + j) \* 4);

color = Xil\_In32(color\_addr);

}

Xil\_Out32(TFT\_FRAME\_ADDR0 + addr, color);

}

**return**;

}

**void** **display\_score**(**int** val){

**int** digits[6] = {0, 0, 0, 0, 0, 0};

**for** (**int** i = 5; i >= 0; i--){

digits[i] = val % 10;

val /= 10;

}

**for** (**int** i = 0; i < 6; i++){

**int** pos\_row\_start = 150 , pos\_col\_start = 5 + i \* 25;

**for** (**int** row = 0; row < 40; row++)

**for** (**int** col = 0; col < 25; ++col){

u32 color = Xil\_In32((**unsigned** **int**)gImage\_digits + 4000 \* digits[i] + ((row \* 25 + col) \* 4));

Xil\_Out32(TFT\_FRAME\_ADDR0 + 4 \* (1024 \* (row + pos\_row\_start) + (col + 470 + pos\_col\_start)), color);

}

}

}

**void** **display\_outcomes**(){

**for** (**int** i = 0; i < 16; i++) display\_bulk(i, maps[i / 4][i % 4]);

}

**void** **display\_message**(**int** occassion) {

**if** (occassion == 0) {

**for** (**int** row = 300; row < 325; row++)

**for** (**int** col = 480; col < 480 + 90; col++) {

u32 color = Xil\_In32((**unsigned** **int**)gImage\_over + (90 \* (row - 300) + (col - 480)) \* 4);

Xil\_Out32(TFT\_FRAME\_ADDR0 + 4 \* (row \* 1024 + col), color);

}

} **else** {

**for** (**int** row = 0; row < 24; row++)

**for** (**int** col = 0; col < 90; col++) {

u32 color = Xil\_In32((**unsigned** **int**)gImage\_win + ((90 \* row + col) \* 4));

Xil\_Out32(TFT\_FRAME\_ADDR0 + 4 \* ((row + 300) \* 1024 + col + 480), color);

}

}

}

Logic.c代码如下所示：  
**#include** "logic.h"

**#include** "stdio.h"

**#include** "stdlib.h"

**#include** "display.h"

**extern** **unsigned** **int** maps[4][4];

**int** total\_score = 0;

**int** directions[4][2] = {{0, 1}, {0, -1}, {-1, 0}, {1, 0}};

**void** **init\_game**(**void**) {

**for** (**int** i = 0; i < 4; i++)

**for** (**int** j = 0; j < 4; j++)

maps[i][j] = 0;

total\_score = 0;

generate\_new();

}

**void** **generate\_new**(){

**int** pos\_empty[16], cnt = 0;

**for** (**int** i = 0; i < 16; i++)

**if** (maps[i / 4][i % 4] == 0) {

pos\_empty[cnt] = i;

cnt++;

}

**srand**(**rand**());

xil\_printf("random :%d\n", cnt);

**int** select\_pos = pos\_empty[**rand**() % cnt];

maps[select\_pos / 4][select\_pos % 4] = 2;

xil\_printf("random :%d %d\n", select\_pos / 4, select\_pos % 4);

display\_outcomes();

}

**void** **move\_forward**(**int** id){

**if** (id == 1) {

**for** (**int** col = 0; col < 4; col++) {

**int** ans[4] = {0}, cnt = 0, lst = -1;

**for** (**int** i = 0; i < 4; i++) {

**if** (maps[i][col] == 0) **continue**;

**if** (lst == -1) {

ans[cnt++] = maps[i][col];

lst = i;

} **else** **if** (maps[i][col] == maps[lst][col]){

ans[cnt - 1] \*= 2;

total\_score += ans[cnt - 1];

lst = -1;

} **else** {

ans[cnt++] = maps[i][col];

lst = i;

}

}

**for** (**int** i = 0; i < 4; i++)

maps[i][col] = ans[i];

}

display\_score(total\_score);

generate\_new();

} **else** **if** (id == 2) {

**for** (**int** col = 0; col < 4; col++) {

**int** ans[4] = {0}, cnt = 3, lst = -1;

**for** (**int** i = 3; i >= 0; i--) {

**if** (maps[i][col] == 0) **continue**;

**if** (lst == -1) {

ans[cnt--] = maps[i][col];

lst = i;

} **else** **if** (maps[i][col] == maps[lst][col]){

ans[cnt + 1] \*= 2;

total\_score += ans[cnt + 1];

lst = -1;

} **else** {

ans[cnt--] = maps[i][col];

lst = i;

}

}

**for** (**int** i = 3; i >= 0; i--)

maps[i][col] = ans[i];

}

display\_score(total\_score);

generate\_new();

} **else** **if** (id == 3){

**for** (**int** row = 0; row < 4; row++) {

**int** ans[4] = {0}, cnt = 0, lst = -1;

**for** (**int** i = 0; i < 4; i++) {

**if** (maps[row][i] == 0) **continue**;

**if** (lst == -1) {

ans[cnt++] = maps[row][i];

lst = i;

} **else** **if** (maps[row][i] == maps[row][lst]){

ans[cnt - 1] \*= 2;

total\_score += ans[cnt - 1];

lst = -1;

} **else** {

ans[cnt++] = maps[row][i];

lst = i;

}

}

**for** (**int** i = 0; i < 4; i++)

maps[row][i] = ans[i];

}

display\_score(total\_score);

generate\_new();

} **else** **if** (id == 4) {

**for** (**int** row = 0; row < 4; row++) {

**int** ans[4] = {0}, cnt = 3, lst = -1;

**for** (**int** i = 3; i >= 0; i--) {

**if** (maps[row][i] == 0) **continue**;

**if** (lst == -1) {

ans[cnt--] = maps[row][i];

lst = i;

} **else** **if** (maps[row][i] == maps[row][lst]){

ans[cnt + 1] \*= 2;

total\_score += ans[cnt + 1];

lst = -1;

} **else** {

ans[cnt--] = maps[row][i];

lst = i;

}

}

**for** (**int** i = 3; i >= 0; i--)

maps[row][i] = ans[i];

}

display\_score(total\_score);

generate\_new();

}

**if** (check\_lose()) display\_message(0);

**if** (check\_win()) display\_message(1);

}

**int** **check\_lose**(**void**){

// for (int col = 0; col < 4; col++) if (maps[0][col] == 0) return 0;

**for** (**int** i = 0; i < 4; i++)

**for** (**int** j = 0; j < 4; j++){

**if** (maps[i][j]) {

**for** (**int** k = 0; k < 4; k++){

**int** new\_x = i + directions[k][0], new\_y = j + directions[k][1];

**if** (new\_x < 0 || new\_x > 3 || new\_y < 0 || new\_y > 3) **continue**;

**if** (maps[new\_x][new\_y] == maps[i][j]) **return** 0;

}

} **else** **return** 0;

}

**return** 1;

}

**int** **check\_win**(**void**) {

/\*\*

\* check whether the maps are full or has encountered 2048

\* if the answer is yes, return 1

\* else return 0.

\*/

**for** (**int** i = 0; i < 4; i++)

**for** (**int** j = 0; j < 4; j++){

**if** (maps[i][j] == 256) **return** 1;

}

**return** 0;

}

Main.c的代码如下所示：

**#include** "xintc\_l.h"

**#include** "xgpio\_l.h"

**#include** "xparameters.h"

**#include** "xio.h"

**#include** "xil\_exception.h"

**#include** "display.h"

**#include** "logic.h"

**#define** TFT\_BASEADDR 0x44a00000

**#define** TFT\_FRAME\_BASEADDR 0x80000000

**unsigned** **int** maps[4][4] = {0};

**int** total\_scores = 0;

**int** **main**(){

init\_display();

init\_game();

Xil\_Out32(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_TRI\_OFFSET, 0x1f);

**unsigned** **short** last\_status = 0;

**while** (1) {

// loop for reading the statues from the buttons.

**for**(**int** i = 0; i < 100000; i++);

**unsigned** **short** status = 0;

status = Xil\_In16(XPAR\_AXI\_GPIO\_0\_BASEADDR + XGPIO\_DATA\_OFFSET) & 0x1f;

**if** (status != last\_status) {

last\_status = status;

} **else** **continue**;

**int** direction = 0;

**switch** (status) {

**case** 0x2:

direction = 1;

**break**;

**case** 0x10:

direction = 2;

**break**;

**case** 0x4:

direction = 3;

**break**;

**case** 0x8:

direction = 4;

**break**;

**case** 0x1:

init\_display();

init\_game();

**break**;

**default**:

**continue**;

}

move\_forward(direction);

}

**return** 0;

}