技术文件

完成时间：2015年1月3日

智能系统设计

实验报告

**设计小组编号：16**

**设计项目名称：智能系统设计**

**设计小组名单：韩迪（组长）、许校培**

上海交通大学 电子信息与电气工程学院

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 班级 | 学号 | 具体负责的工作 |
| 韩迪 | F1103019 | 5110309570 | 内核搭建、报告撰写 |
| 许校培 | F1103006 | 5110309078 | 人工智能算法设计、报告撰写 |

**摘 要：**

本文介绍了采用Nexys-3 FPGA/SOPC 实验板和外设的VGA显示器开发的五子棋游戏。通过把FPGA内嵌的Microblaze 软核CPU应用到游戏中，实现对游戏的控制功能。我们采用Verilog语言编写和修改内核，实现了FPGA对VGA显示器的驱动控制。采用C语言编写游戏界面和算法，实现人人对战、人机对战和机机对战等功能。最后对游戏进行了测试，测试结果验证了本系统方案设计的正确性和基于Microblaze 软核CPU开发游戏的可行性。

**关键词：**

人工智能、五子棋、FPGA、估值函数

**ABSTRACT:**

This article introduce the development of a gobang game based on the Nexys - 3 FPGA/SOPC experiment board and peripherals, such as VGA display. The FPGA embedded Microblaze soft-core CPU is applied to the game to realize the control function. We used Verilog language to write and modify the kernel, and realized that the FPGA can control the VGA display. C language is used to program the game’s interface and algorithm, and to realize the function of Human-Computer fighting, Human-Human fighting and Computer simulation. Finally, the game was tested, the results validate the correctness of the system design and the feasibility of game development based on Microblaze soft-core CPU.

**KEYWORDS:**

Artificial intelligence, Renju(Gobang), FPGA、estimate function

目 录

[**1. 概述** 3](#_Toc408083669)

[1.1 编写说明 3](#_Toc408083670)

[1.2 缩略语 3](#_Toc408083671)

[**2. 系统总体说明** 4](#_Toc408083672)

[2.1 课题任务规定的设计要求 4](#_Toc408083673)

[2.2 实际完成后的功能 4](#_Toc408083674)

[2.3 系统的设计原理与分析 4](#_Toc408083675)

[**3. 系统的硬件结构** 6](#_Toc408083676)

[3.1 硬件总体结构 6](#_Toc408083677)

[3.2 VGA IP CORE描述 6](#_Toc408083678)

[**4. 系统算法部分** 12](#_Toc408083679)

[4.1 基本的评分规则及实现 12](#_Toc408083680)

[4.2 进攻和防守——局面评估函数 12](#_Toc408083681)

[4.3 博弈树与极大极小搜索 13](#_Toc408083682)

[4.4 α-β剪枝以及修正α-β剪枝法 14](#_Toc408083683)

[4.5 禁手规则 16](#_Toc408083684)

[**5. 系统功能及测试** 17](#_Toc408083685)

[5.1 测试结果 17](#_Toc408083686)

[**6. 致谢与感想** 19](#_Toc408083687)

[**7. 参考资料** 20](#_Toc408083688)

# 概述

## 编写说明

本文档主要内容包括使用Xilinx Platform Studio（简称XPS）开发工具设计适用于Nexys-3 FPGA/SOPC 实验板和外设的VGA显示器的五子棋游戏，搭设VGA端口输出平台的硬件设计部分，和使用Xilinx Software Development Kit（简称SDK）在已有平台上编写五子棋人人对战、人机对战的软件设计部分。

本文档适合FPGA开发和五子棋算法爱好者阅读，由于笔者能力水平有限，文档不免出现错误，欢迎批评指正。

## 缩略语

FPGA: Field－Programmable Gate Array

XPS：Xilinx Platform Studio

SDK：Xilinx Software Development Kit

VGA：Video Graphics Array

# 系统总体说明

## 课题任务规定的设计要求

关于显示界面：棋盘完整、圆子落于棋盘交叉线、菜单功能齐全美观且操作方便、可以显示输赢、可以重新对弈、可以悔棋、有禁手规则；

关于对弈过程：实现人人对战、人机对战、机机对战。

## 实际完成后的功能

关于显示界面：棋盘完整、圆子落于棋盘交叉线、菜单功能齐全美观且操作方便、可以显示输赢、可以重新对弈、可以悔棋、有禁手规则、精美的悔棋和计时动画；

关于对弈过程：实现人人对战、人机对战、机机对战、先后手选择功能、串口通信显示落子和输赢等信息。

## 系统的设计原理与分析

### 系统的设计原理和总体结构

设计原理通过Nexys-3 FPGA的USB接口外接键盘实现输入控制，通过VGA接口外接VGA显示器实现显示游戏界面。软件设计总体上有人人模式和人机模式两种模式切换。人人对战模式中主要写了键盘输入控制和重新开局、悔棋、禁手、判断胜负等等功能；人机模式中添加了αβ剪枝算法和估值函数来对当前棋盘进行评判，实现计算机的智能输出。

整个软硬件开发的工作流程。从ISE基本设计输入开始，包括了XPS硬件系统架构的生成，再到SDK的软件开发，同时在ISE完成综合、约束、映射等编译步骤，最终生成可下载文件进行板级的调试验证。

接下来，我们使用XPS添加一个带MicroBlaze软核的嵌入式系统。在XPS中进行了相关的配置后便生成了Microblaze的软核。之后将开发平台转移到SDK上，进行基于C语言的程序设计。

在建立好了MicroBlaze软核后，就可以在SDK的环境中使用C语言编写程序了，本次设计中的五子棋智能算法就是在该环境下使用C语言编写出来的，具体说明见后文。在生成IP核时可以自动生成PS/2键盘的驱动和VGA显示器的驱动，同时，本设计中为了实现棋盘的图形化显示和字符显示，我们在user\_logic.v驱动文件中使用Verilog语言进行了编写和修改。

### 系统运行流程简介

当进入游戏系统之后，可以通过空格键（Space键）在人人对战和人机对战间切换，且可以选择先后手。进入游戏后，两方轮流通过键盘输入或者通过系统的五子棋人工智能算法选择落子点，之后系统更新当前棋盘的状态，并将该棋盘状态显示在VGA显示器上。完成显示后，将进行禁手判断，黑方若出现禁手则直接判定为输，否则进入常规输赢判定流程。此时若无任何一方满足赢棋判定条件，则交换落子方，否则显示胜负结果。在任何一方落子之后，开启时长30秒的倒计时，如果超时则判断该方放弃此次落子机会，自动改为另一方落子。同时可以分别通过R键和ESC键选择悔棋和重新开局。

# 系统的硬件结构

## 硬件总体结构

本设计采用Nexys-3 FPGA/SOPC 实验板和MicroBlaze软核实现硬件和软件的协同设计。整个设计可以分为三个部分:FPGA部分、存储器部分及VGA接口。在FPGA中内建了一个Nios II嵌入式处理器和一个支持标准VGA接口的VGA控制器；存储器部分由外接的SRAM和Flash来构成，在Flash中放置了游戏的程序代码，而SRAM用于程序的运行和数据处理。VGA接口部分提供了一个15针的标准VGA接口。本设计中构造的EP2C35F672C6模块包括已经封装的构建NIOS II Processor、FlashMemory、Intervaltimer、JTAG UART、PIO(ParallelUO)、SRAM、DMA和Avalon Tristate Bridge以及根据自己的需要自定义的VGA控制器；五子棋游戏系统的微处理器可通过SOPC Builder中的微处理器配置来实现。

本系统用到的硬件配置见图3.1所示。

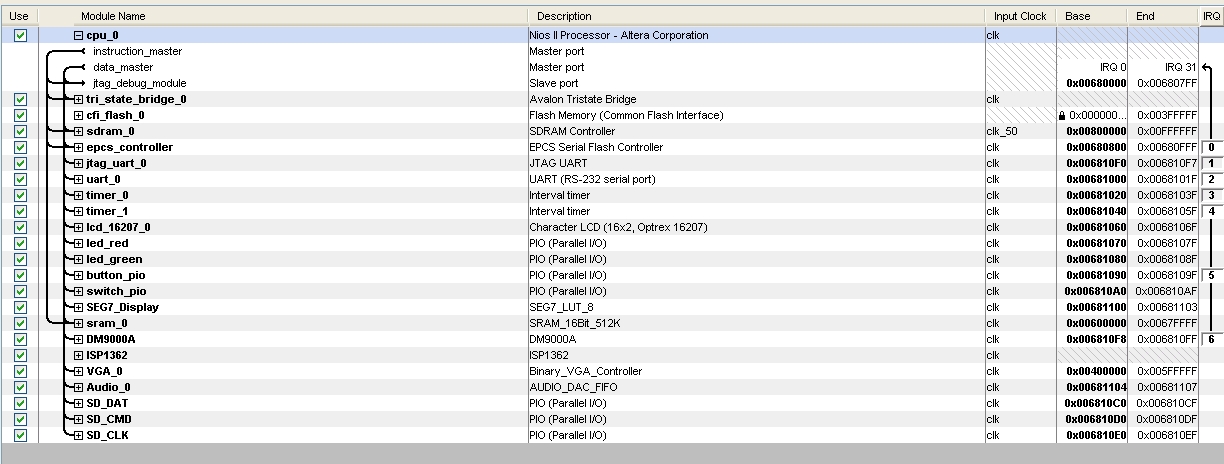


图3.1 系统硬件配置图

在提供了相应组件的IP核后就可以在SOPC Builder中进行配置。当在SOPC Builder中加入了所有的组件并指定相应的系统参数后，SOPC Builder将产生Avalon总线模块，输出描述系统的HDL文件和相关文件，之后就可以进行软件的设计。

## VGA IP CORE描述

### 功能描述

在Xilinx的EDK下使用VGA端口输出，最简单的方法是使用Xilinx官方提供的TFT的IPCore，该IPcore包含了现成的硬件模块和API，使用简单方便。但TFT需要2MB的RAM作为缓冲区，本次实验所使用的Nexys3开发板的片内RAM容量不够，片外Celluar RAM速度又不能满足要求，因此这里无法使用TFT的IPcore。这里我们使用了老师给出的VGA\_IP CORE参考资料[1]，实现了五子棋输出的VGA输出功能。

### 实现方式[1]

1. 在Xilinx Platform Studio下新建一个基于Microblaze的硬件系统，并选择Create or Import Peripheral Wizard新建一个IP，具体流程可参考Xilinx大学计划的EDK教程中的lab1~4（见附录2）。我选择的总线类型是PLB，暂不清楚AXI总线的使用方法。此外由于个人比较习惯使用verilog语言，因此在新建IP的最后一步选择了”Generate stub ‘user logic’ template in Verilog instead of VHDL”。设新建IP名为vga\_ip。
2. 修改vga\_ip的MPD文件，增加自设端口CLK\_25M、VGA\_R、VGA\_G、VGA\_B、VGA\_HSYNC和VGA\_VSYNC如图3.2所示。注意图中的LED端口仅为我调试时所用。

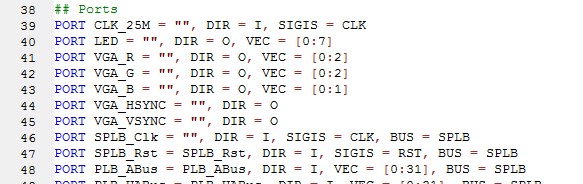


图3.2 修改vga\_ip MPD文件[1]

修改vga\_ip源文件中的vga\_ip.vhd文件。注意虽然在设计时选择了verilog语言，但vga\_ip.vhd文件作为IP的封装和接口，总是用VHDL写的。这里只需要加入自定义的输入输出端口即可，自定义的逻辑将在user\_logic.v文件中加入。

vga\_ip.vhd文件中需要修改的地方共有三处，分别如图2~4所示。其中，图3.3表示vga\_ip顶层的自定义端口，与MPD文件中进行的修改所对应。图3.4表示对user\_logic.v文件中的端口进行声明，图3.5表示vga\_ip.vhd文件中的端口到user\_logic.v的端口映射。｀

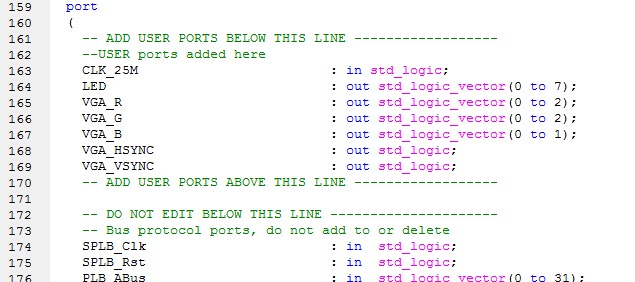


图3.3 自定义vga\_ip端口[1]

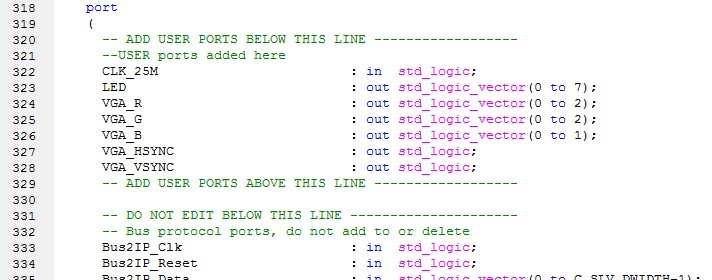


图3.4 自定义user\_logic端口[1]

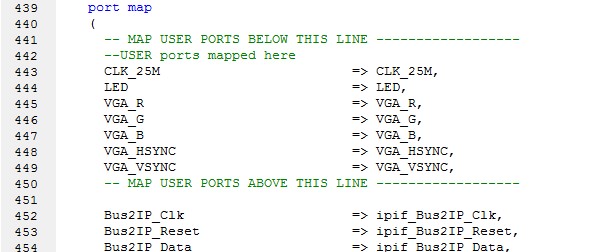


图3.5 自定义端口映射[1]

修改vga\_ip源文件中的user\_logic.v文件。首先进行user\_logic模块的端口定义，如图6所示。这里的定义应与图3.4一致。

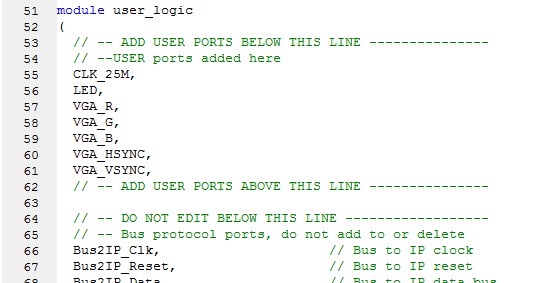


图3.7 自定义user\_logic端口[1]

然后编写适用于五子棋游戏的VGA输出。VGA输出原理较简单，只需理清同步信号VGA\_HSYNC和VGA\_VSYNC的时序关系即可，不再赘述。我们首要解决的，是如何使用较少的资源和较简单的软硬件接口实现棋盘输出的问题。

这里我们组借鉴老师提供的VGA接口程序中的采用了降低分辨率的方法，将棋盘划分为15\*15个颜色块，对应于15\*15个可下棋子的位置。这样只需要划出深度为225的寄存器组，即可表示五子棋的棋盘。每个色块由32\*32个像素构成，即每32\*32个像素作为一个最小单位来处理。这样整个棋盘的大小以像素为单位，为480\*480。

对于这225个数据，为了进一步减少储存容量的要求，每个数据只用两位的宽度来表示其类型，一共可以有4个类型。每个数据通过一个简单的映射关系对应到八位VGA颜色数据。具体地，若数据类型为0，则表示该数据应显示的是一方棋子，类型为1，表示为另一方棋子，类型为2表示为棋盘的背景色，类型为3表示为光标。

综上，棋盘的存储可由下式定义：

reg [0 : 1] board[0 : 225-1];

同理，为了实现屏幕右侧的时间显示和菜单界面，将右侧界面160\*480的像素分割为300个16\*16的小单位，故该部分的存储定义为：

reg [0 : 1] menu[0 : 300-1];

Microblaze的数据总线宽度为32位，为了充分合理的利用总线宽度，定义[0 : 7]位表示要修改的色块的横坐标，[8 : 15]位表示要修改的色块的纵坐标，两者的实际取值范围均为0~15。[24 : 31]位为该单位的类型，如上文所述应为0~3中的一种，对应黑白棋子、棋盘或光标。

那么，从总线接收数据、改变VGA显示内容的逻辑为：

always@(\*)

begin

if (vga\_ena == 1)

begin

if (x\_cor < 481)

begin

if((x\_cor%32-16)\*(x\_cor%32-16)+(y\_cor%32-16)\*(y\_cor%32-16)<196 && board[x\_cor / 32 + y\_cor / 32 \* 15]!=2)

vga\_data <= colormap[board[x\_cor / 32 + y\_cor / 32 \* 15]];

else if(x\_cor[5 : 9]==5'b10000 || y\_cor[5 : 9]==5'b10000 || x\_cor==480)

vga\_data <= linecolor;

else vga\_data <= 8'b11110100;;

end

else

vga\_data <= colormap[menu[(x\_cor - 480) / 16 + y\_cor / 16 \* 10]];

end

else

vga\_data <= 0;

end

代码中,x\_cor和y\_cor为扫描像素的横纵坐标，vga\_ena为VGA输出的使能端，vga\_data为输出的八位VGA数据。寄存器组colormap负责将棋盘状态board[]转换为VGA数据。

为了显示棋盘上划分色块的黑线，可判断x\_cor与y\_cor的后五位是否是二进制形式下的10000，亦即当前横纵坐标是否能被32整除后余16。

同时为了让五子棋游戏的棋子以圆点的形式显示在棋盘的交叉线上，通过((x\_cor%32-16)\*(x\_cor%32-16)+(y\_cor%32-16)\*(y\_cor%32-16)<196语句判断，将离落子处距离小于或等于14像素的位置点亮，形成圆形的棋子。

棋盘的实际表现形式可见图3.8。

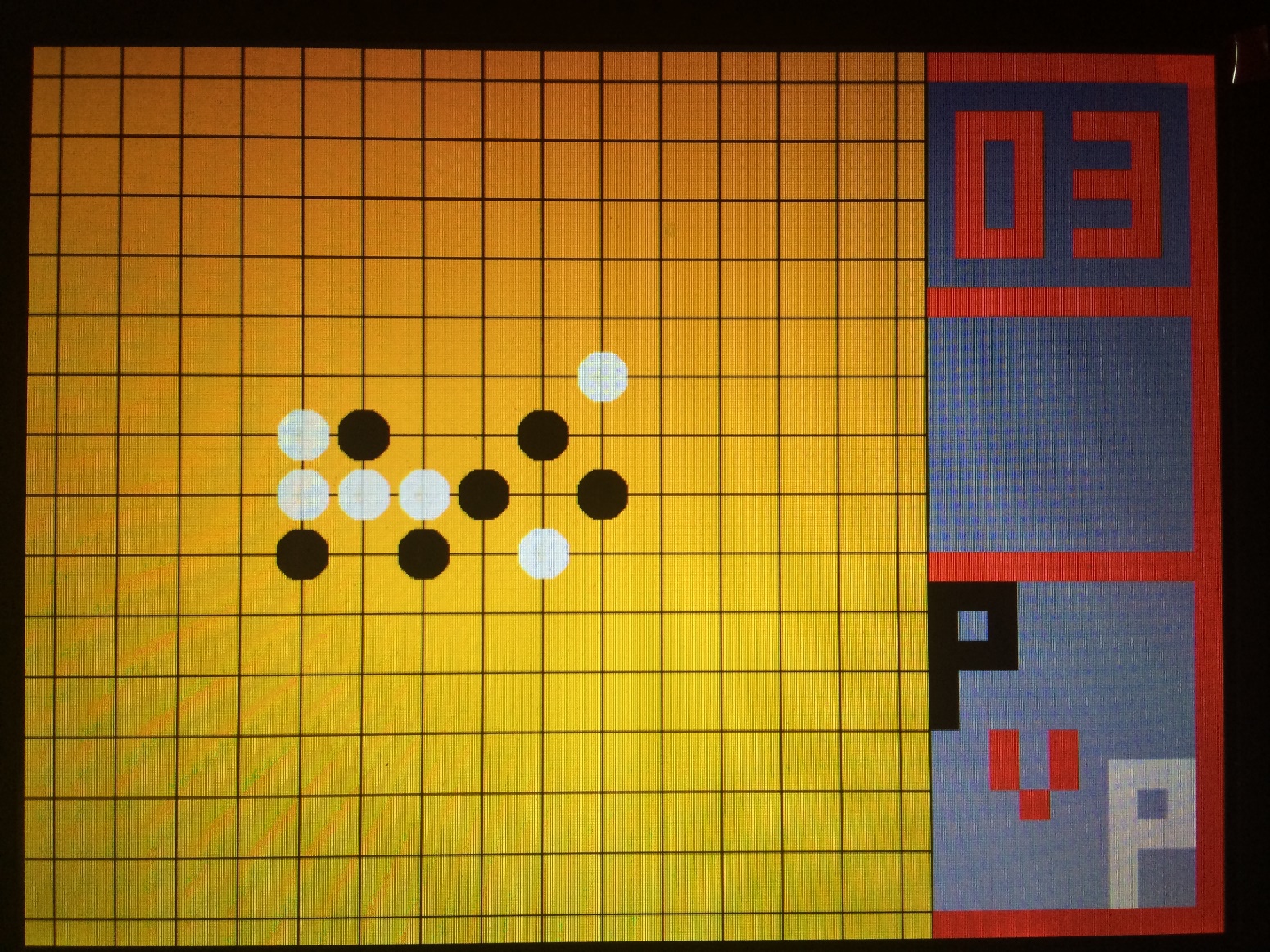


图3.8 棋盘的表现示意图[1]

1. 完成了vga\_ip的设计后，最后再在ucf文件中进行管脚约束即可。这里的管脚应与图1中的端口一致。见图3.9。

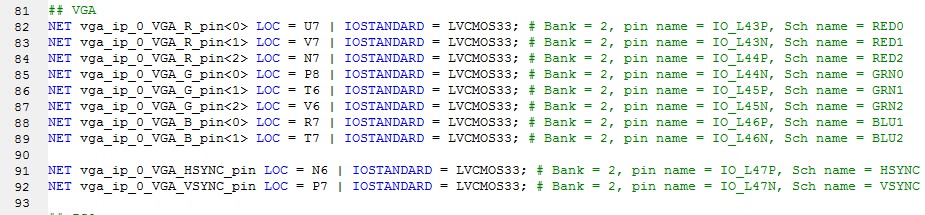


图3.9 修改ucf文件[1]

# 系统算法部分

本节将介绍游戏的算法部分的实现，即人机对战中不同级别AI的实现。

## 基本的评分规则及实现

五子棋的AI算法有很多种类，我们这里采用最普遍的算法，即**局面评估-搜索**算法，以构成能和人类博弈的AI。对于五子棋来说，胜负判定的规则是如果一方形成五连（或者更多连子），则判定该方获胜（关于禁手规则将在4.5小节中介绍）。而对于形成五连，有许多固定的棋型，例如活四，死四，活三，死三等等，这些棋型对于场面的胜负天平的贡献各不相同，因此便可以设定一个评分函数，对这些棋型进行评分，在这里，我们综合各方面资料，使用的评分函数如表4-1：

表4-1 特定棋型评分函数[2]



有了评分函数，则可以对某一次落子产生的效果进行一定的评估，显然棋型都是一维的，所以可以得到**棋型评估函数**，用以评估某一行上某一种颜色棋子的得分情况。

## 进攻和防守——局面评估函数

有了**棋型评估函数**，显然并不能进行搜索选择，还需要对整个棋盘上的局面进行评估，这就涉及到了**局面评估函数**，即将棋盘上所有的特殊棋型的得分进行判定和统计。具体的实现是按照每一行，每一列，每一斜行（左斜，右斜）的进行计算和加和。对于一个15\*15的棋盘来说，一共是15+15+29+29=88行， 对这88行的棋型进行估值（调用**棋型评估函数**）和求和，得到总体评分（但是对五子棋来说至少要5以上连子才能获得胜利，所以再减去四个角落的16行，只需计算72行）。

对于电脑来说，似乎只需要知道电脑所执子的**局面评估函数**得分然后再进行判定即可，但这样面临一个问题，就是忽略了防守，特别在人执黑先行的时候，AI不会再理会玩家的落子，而是自己全力进攻，这样AI达到的效果并不理想，因此还需要完善**局面评估函数。**

完善的方法较为简单，即将AI一方的评估总分设为负值，将再对玩家一方的局面进行评估，设为正值，将两者进行加和，便是整个局面的完整**局面评估函数**。可知得分越小（越偏向于负值）对AI越有利，得分越大（越偏向于正值）对玩家越有利。

此时可以形成最简单的初级难度算法，即将对于所有可下子的点，进行假定下子之后的局面评估，得出分数即为该点得分，然后再进行挑选（对于AI挑选得分最小的点），对于权值一样的点，随机进行挑选，即可得到AI落子位置。

## 博弈树与极大极小搜索

4.2节所介绍的初级难度算法实际上基于贪婪法实现，但这样的算法缺陷在于不能对未来局面进行判定，因此智能程度不高。因此更高级的算法需要对未来的局面进行评估。

以甲乙两人对弈五子棋为例。假定现在该甲走棋, 且甲有若干种走法, 而对甲的任一走法, 乙也可以有与之对应的不同的多种走法, 而对乙的走法甲又有若干种方法应对, 如此反复。显然, 若从当前棋局状态(根节点)出发,找出所有可能的乙的走法(子节点), 再从每个子节点出发找出甲对应于每个乙的走法的所有应对(子子节点), 如此反复直到终局。由此构成了一棵树。[3]

从理论上来说，对于一个有限位置的棋盘，可以找出一颗博弈树包含所有情况，但这样的博弈树太过庞大，其计算量之大是目前所有的计算机都无法承受的, 因此,用博弈树法来解决人机博弈时,通常只能搜索到一个非常有限的深度,并根据此有限深度的形势来判断每种走法的优劣。

博弈树的实现是在之前估值函数的基础上，增加1层或多层的预测。简单说来就是不再是直接的贪婪法，而是还要考虑另一方的下子结果。这种考虑使用的便是极大极小搜索。如图4.1，设若该甲方下子（甲方为正，乙方为负），那么甲方目前可以有第二层节点的三种下子方式，而对于甲方的三种下子方式，乙方又有不同的应对，这些应对方式根据不同的局面评估函数可以得到不同的分数，自然对于甲方的不同落子，乙方肯定会选择使分数最低的那个应对方式，因此可见对于甲方第二层的三种方式，在乙方应对之后，得到的分数为，300，-400和-300。

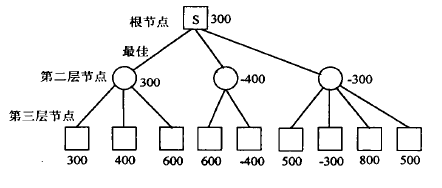


图4.1 博弈树示例

此时我们便可以在第二层中选出得分最大（有利于甲）的选择，即第一个（得分为300）的落子方式。

这样的预测可以达到很大的数目，在中级算法中我们只用了两层博弈树，达到了提升难度的效果。

## α-β剪枝以及修正α-β剪枝法

当把博弈树的层数增加时，硬件的计算性能便明显不够了，此时需要对于算法进行优化，以减少计算量，使用α-β剪枝可以有效的去除博弈树极大极小搜索中很多不必要的计算。

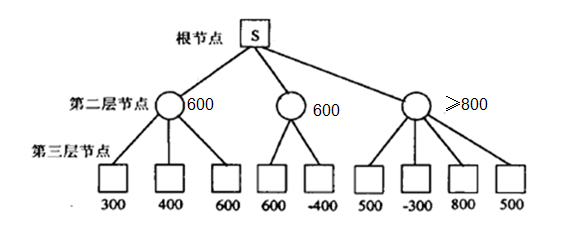


图4.2 α-β剪枝示意图

如图4.2，S点为根节点，假设现在该AI下子，那么在没有剪枝之前，AI会遍历所有的点，然后对于每一个点，在下子之后，再遍历所有的对方下子的点，从中找到极大极小值。这就达到了O（n^2），但采取α-β剪枝后可以极大的减少运算量。

我们从第二层第一个节点开始，如果AI下在这里，那么，到了第三层就是人开始下的棋，按照贪婪法，人必然会下使整个局面评分最大的那一点，也就是第三层第三点，得分为600，那么对于AI来说，这一点的估值便只能是600（不能期待对方会按利于我方的方式下棋）。

那么再看第二层下一点，由刚才的流程可以想见，这一点的最终估值还是600

此时注意，AI目前所能得到的最小值是600

那么对于第二层第三点，可以看到，对于这个点的子节点，第1,2点都是小于600的，但是现在计算到第3个点，800，由此，对方必然会选择这个点或者乃至更大的大于800的点，因此第二层第三个点的值必然大于等于800.

到这里，我们便不用再向下计算了。因为第二层这个点肯定是不可能选择的，因此整个这一支后面就不需要再计算下去，这就形成了剪枝。

α-β剪枝可以有效的降低计算量，同时我们再进行一定提升，使用修正α-β剪枝法还可以再优化算法。

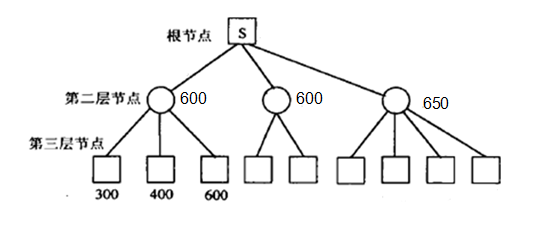


图4.3 修正α-β剪枝示意图

如图4.3所示，在我们计算完某一支以后，得到了目前的最小权值点600，然后我们对第二层下一个点进行计算，此时还没有预测对方下子的情况，直接得到的估值函数便是600。 那么可以想见，此时轮到对方下子，一旦下子，局面估分必然会增长，也就是说无论对方怎么下子，最后该点得分必然大于600，因此该枝不需要继续计算。

同理后面一点直接计算便比当前最小值600更大，那么也不需要再计算。

这样我们在没有进行第三层计算的情况下减去了两支，极大地减小了计算量。

通过剪枝算法，便可以在使用更高层的博弈树，我们的高级难度便是使用了三层博弈树。

## 禁手规则

在五子棋中，先行黑子有必胜优势，因此需加以禁手限制，规则为：

**所谓黑方形成禁手，是指黑方一子落下同时形成两个或两个以上的活三、冲四或长连禁手。**

如此便可以简单地根据定义进行判定。禁手判定算法是统计落子之前之后活三，冲四等的数量，例如对于AI一方，下子之前有1个活三，下子之后有3个活三，那么判定为禁手，冲四同样。而对于长连，只要先手形成连6即判定为长连。

对于AI来说，禁手判定会直接将该点的估值调到极高，因此AI不会选择在该点落子。而一旦落子后被判定禁手，则会直接判为失败。

# 系统功能及测试

## 测试结果

系统最终测试效果如图5.1所示。用户可以通过PS/2接口键盘进行输入，五子棋的棋盘和用户交互信息都输出在VGA显示器上，而所有的计算处理都在Nexys3板卡的FPGA上进行。用户可以通过按键移动棋子，可以进行五子棋人人、人机对战。

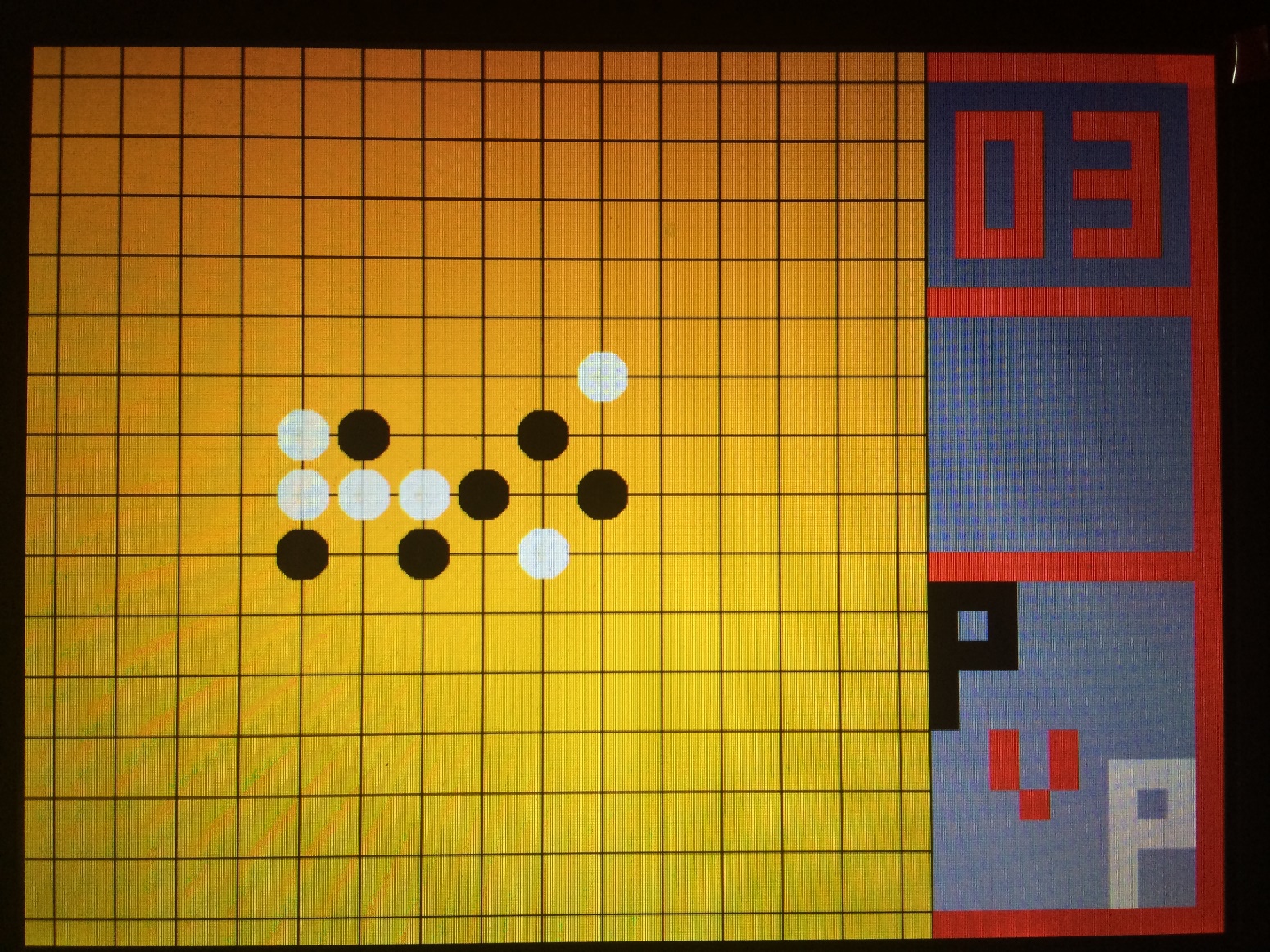


图5.1 游戏效果

可以通过Nexys3板卡上的UART接口，连接上电脑，通过SSCOM串口工具和FPGA中编写的程序，完成FPGA和电脑之间的通信。FPGA可以实时将落子位置，输赢判断以及计时情况等信息发送给电脑，并在SSCOM串口工具上显示。SSCOM串口工具界面如图5.2所示。



图5.2 UART串口通信

# 致谢与感想

智能系统这门课程放在大四上这个学期显然有其意义所在。在大学的前三年中，在程序设计思想与实现课程中，我们初步接触了编程思想，学习了包括Python和C＋＋在内的高级编程语言的基础知识；在数据结构课程中，我们了解了各种算法，学会了从空间和时间两方面的效率来分析算法的思想；而在VHDL课程中，我们初步接触了硬件编程语言，学习了面向硬件的变成理念。最终，在智能系统设计这门课中，我们第一次同时运用到了这几门专业课，将各个科目融会贯通并且第一次应用于实践课程中，算得上时对大学前三年学习成果的一次检验和总结。

在做项目的过程中，会遇到各种各样的困难和弯路，组员也不可避免地会犯下各种各样的错误。在这种情况下，最重要的是共同努力把问题解决，相互交流，而不应该相互指责埋怨。在大四上这个人生的紧要关口，本组成员都在为自己未来何去何从而拼搏和忙碌，有人忙着出国申请和英语考试，有人忙着准备外推和毕设，并没有因为是大四学年而感到轻松和空闲，所以组员之间的相互体谅与相互关怀就显得尤为重要。

同时，要善于向别人请教，有些问题并不难，关键是有没有想到点子上或者是没有经验而走了弯路，所以及时向别人请教可以节省大量的时间。这点我们在这次实验过程中深有体会。在召开的两次五子棋组的交流会上，杨宇红老师一直给我们提供思路，并积极回答大家的提问，给予了我们组很大的帮助；同时也要感谢缪瑞航和覃瑶同学，每次的互相交流、互相讨论都为彼此提供了新的想法和思路，并愿意分享自己的经验。在其他组的同学遇到了什么困难，我们也会提供建议和力所能及的帮助。在这门课中，相互合作和互相帮助这样的精神得到了极大的体现。

最后还有一点建议，就是希望老师能够演示内核搭建和开发工具使用的方法和步骤，或者提供较为详细的说明资料。本次课程的重点应该是人工智能算法的设计，但是因为不熟悉软件使用和硬件开发，在这一块我们组和其他小组花费了大量的时间也没有完成解决问题，最后也是在别组帮助下学会了开发工具的使用。

# 参考资料

[1] VGA接口的FPGA代码\_Nexys3，上海交大电子工程系，ftp://202.120.39.248

[2]王长飞,蔡强,李海生. 智能五子棋算法的设计实现[J]. 系统仿真学报, 2009, (4):1051-1054.

[3]智能系统设计课件1-2014 ，上海交大电子工程系，ftp://202.120.39.248

# 附录：主程序代码清单

#include "xparameters.h"

#include "xgpio.h"

#include "xutil.h"

#include "vga\_ip.h"

#include "xps2.h"

#include <xtmrctr.h>

#include <xintc\_l.h>

#include "lab\_gobang\_AI\_2.h"

#include "VGADraw.h"

#include "AI.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#define Human 1

#define Computer 0

#define Nothing 2

typedef struct node

{

int x;

int y;

}POINT;

int X,Y;

int N=15; //棋盘大小

int huo4[3]; //活四数

int huo3[3]; //活三数

volatile int Total =0 ;//已下棋子总数

int pointValue[15][15]; //每点评分

int pointValue2[15][15]; //博弈树每点评分

int who=Nothing;

volatile int fall\_flag;//人是否落子标志

void everyScore(int who);

//====================================================

int board\_state[15][15];

POINT board\_record[225];//双方落子记录

int r\_times=0;

int sumall(int data[N][N]) ;

int sumline(int line[],int length,int who);

int p\_Score(int num,int numof0, int bp[],int who,int zhong0) ;

int jinshou() ;

POINT current\_pos,previous\_pos;

POINT best(int who) ;

POINT bestboyi(int who);

//======================================================

int huo4[3]={0,0,0}; //活四数

int prehuo4[3]={0,0,0}; //之前的活四数

int huo3[3]={0,0,0}; //活三数

int prehuo3[3]={0,0,0}; //之前的活三数

int changlian=0; //如果人长连禁手，则置为1

int first; //谁先走

int min;

//====================================================

int board\_state[15][15];

POINT board\_record[225];

int turn;

int level=1;

int status=PVP;

int step\_flag;

int win\_flag;

int BackTimes=0;

int maxMoveX, maxMoveY;

volatile int time0=0;

volatile int count=0;

volatile u32 ssBuf;

static const u32 rgfsNumMap[10] = {0x000000C0, 0x000000F9, 0x000000A4,

0x000000B0, 0x00000099, 0x00000092,

0x00000082, 0x000000F8, 0x00000080,

0x00000090};

//====================================================

int CheckWin(int x\_pos, int y\_pos, int turn)

{

int i,j,count;

//水平

count=1;

for (i=x\_pos+1, j=y\_pos; i<15; i++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1, j=y\_pos; i>=0; i--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==5)

return 1;

//垂直

count=1;

for (i=x\_pos, j=y\_pos+1; j<15; j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos, j=y\_pos-1; j>=0; j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==5)

return 1;

//左上至右下

count=1;

for (i=x\_pos+1,j=y\_pos+1; i<15 && j<15; i++,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos-1; i>=0 && j>=0; i--,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==5)

return 1;

//左下至右上

count=1;

for (i=x\_pos+1,j=y\_pos-1; i<15 && j>=0; i++,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos+1; i>=0 && j<15; i--,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==5)

return 1;

//no win

return 0;

}

int CheckBan(int x\_pos, int y\_pos, int turn)

{

int i,j,count,lian3,lian4,lian6;

lian3=0;

lian4=0;

lian6=0;

//水平

count=1;

if(board\_state[x\_pos+1][y\_pos]==EMPTY){

for (i=x\_pos+2, j=y\_pos; i<15; i++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1, j=y\_pos; i>=0; i--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos-1][y\_pos]==EMPTY){

for (i=x\_pos+1, j=y\_pos; i<15; i++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-2, j=y\_pos; i>=0; i--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos-1][y\_pos]!=EMPTY&&board\_state[x\_pos+1][y\_pos]!=EMPTY){

for (i=x\_pos+1, j=y\_pos; i<15; i++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1, j=y\_pos; i>=0; i--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

if (count>5)

++lian6;

}

//垂直

count=1;

if(board\_state[x\_pos][y\_pos+1]==EMPTY){

for (i=x\_pos, j=y\_pos+2; j<15; j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos, j=y\_pos-1; j>=0; j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos][y\_pos-1]==EMPTY){

for (i=x\_pos, j=y\_pos+1; j<15; j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos, j=y\_pos-2; j>=0; j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos][y\_pos+1]!=EMPTY&&board\_state[x\_pos][y\_pos-1]!=EMPTY){

for (i=x\_pos, j=y\_pos+1; j<15; j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos, j=y\_pos-1; j>=0; j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

if (count>5)

++lian6;

}

//左上至右下

count=1;

if(board\_state[x\_pos+1][y\_pos+1]==EMPTY){

for (i=x\_pos+2,j=y\_pos+2; i<15 && j<15; i++,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos-1; i>=0 && j>=0; i--,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos-1][y\_pos-1]==EMPTY){

for (i=x\_pos+1,j=y\_pos+1; i<15 && j<15; i++,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-2,j=y\_pos-2; i>=0 && j>=0; i--,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos+1][y\_pos+1]!=EMPTY&&board\_state[x\_pos-1][y\_pos-1]!=EMPTY){

for (i=x\_pos+1,j=y\_pos+1; i<15 && j<15; i++,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos-1; i>=0 && j>=0; i--,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

if (count>5)

++lian6;

}

//左下至右上

count=1;

if(board\_state[x\_pos+1][y\_pos-1]==EMPTY){

for (i=x\_pos+2,j=y\_pos-2; i<15 && j>=0; i++,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos+1; i>=0 && j<15; i--,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos-1][y\_pos+1]==EMPTY){

for (i=x\_pos+1,j=y\_pos-1; i<15 && j>=0; i++,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-2,j=y\_pos+2; i>=0 && j<15; i--,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

}

count=1;

if(board\_state[x\_pos+1][y\_pos-1]!=EMPTY&&board\_state[x\_pos-1][y\_pos+1]!=EMPTY){

for (i=x\_pos+1,j=y\_pos-1; i<15 && j>=0; i++,j--) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

for (i=x\_pos-1,j=y\_pos+1; i>=0 && j<15; i--,j++) {

if (board\_state[i][j]==turn)

count++;

else

break;

}

if (count==3)

++lian3;

if (count==4)

++lian4;

if (count>5)

++lian6;

}

if (lian3>1) return 1;

if (lian4>1) return 1;

if (lian6>0) return 1;

return 0;

}

void timer\_int\_handler(void \* baseaddr\_p) {

unsigned int csr;

int num;

unsigned int MskAnodes;

csr = XTmrCtr\_GetControlStatusReg(XPAR\_XPS\_TIMER\_0\_BASEADDR, 0);

if (csr & XTC\_CSR\_INT\_OCCURED\_MASK && win\_flag==0) {

count++;

if (count%2==0) {

MskAnodes=0x00000D00;

num=(TIME\_LIMIT-time0)/10;

}

else {

MskAnodes=0x00000E00;

num=(TIME\_LIMIT-time0)%10;

}

ssBuf = MskAnodes | rgfsNumMap[num];

Xil\_Out32(XPAR\_SEVSEG\_DISP\_12BITS\_BASEADDR, ssBuf);

}

if (count==200) {

count=0;

time0++;

DrawBack(3,9,EMPTY);

DrawTime(88, EMPTY);

if (time0>20)

DrawTime(TIME\_LIMIT-time0, CURSOR);

else

DrawTime(TIME\_LIMIT-time0, turn);

}

if (time0>TIME\_LIMIT) {

time0=0;

if (turn==HUMAN\_PLAYER) {

turn=COMPUTER\_PLAYER;

xil\_printf("\r\nComputer Player's turn!\r\n");

}

else {

turn=HUMAN\_PLAYER;

xil\_printf("\r\nHuman Player's turn!\r\n");

}

}

XTmrCtr\_SetControlStatusReg(XPAR\_XPS\_TIMER\_0\_BASEADDR, 0, csr);

}

void InitializeGame(int x\_cur, int y\_cur)

{

int i,j;

for (i=0; i<15; i++) {

for (j=0; j<15; j++) {

board\_state[i][j]=EMPTY;

}

}

DrawNumber(level,6,18,EMPTY);

DrawWinning(0,1, EMPTY);

DrawStatus(0, 18, EMPTY,PVP);

DrawBack(3,9,EMPTY);

count=0;

time0=0;

turn=HUMAN\_PLAYER;

step\_flag=0;

win\_flag=0;

DrawBoard();

DrawChess(x\_cur, y\_cur, CURSOR);

DrawTime(88, EMPTY);

DrawTime(30, turn);

DrawStatus(0, 18, COMPUTER\_PLAYER,PVP);

xil\_printf("\r\nGame Start!\r\n");

}

int main (void)

{

XGpio dip;

int dip\_check;

static XPs2 Ps2Inst;

XPs2\_Config \*ConfigPtr;

u32 StatusReg;

u32 BytesReceived;

u8 RxBuffer;

int key\_count=0;

int i;

status=PVP;

int x\_cur=7, y\_cur=7, x\_pos=0, y\_pos=0;

XGpio\_Initialize(&dip, XPAR\_DIP\_SWITCHES\_8BITS\_DEVICE\_ID);

XGpio\_SetDataDirection(&dip, 1, 0xffffffff);

ConfigPtr = XPs2\_LookupConfig(XPAR\_XPS\_PS2\_0\_0\_DEVICE\_ID);

XPs2\_CfgInitialize(&Ps2Inst, ConfigPtr, ConfigPtr->BaseAddress);

XIntc\_RegisterHandler(XPAR\_XPS\_INTC\_0\_BASEADDR,

XPAR\_XPS\_INTC\_0\_XPS\_TIMER\_0\_INTERRUPT\_INTR,

(XInterruptHandler) timer\_int\_handler,

(void \*)XPAR\_XPS\_TIMER\_0\_BASEADDR);

XIntc\_MasterEnable(XPAR\_XPS\_INTC\_0\_BASEADDR);

XIntc\_EnableIntr(XPAR\_XPS\_INTC\_0\_BASEADDR, 0x1);

XTmrCtr\_SetLoadReg(XPAR\_XPS\_TIMER\_0\_BASEADDR, 0, 333333);

XTmrCtr\_SetControlStatusReg(XPAR\_XPS\_TIMER\_0\_BASEADDR, 0, XTC\_CSR\_INT\_OCCURED\_MASK | XTC\_CSR\_LOAD\_MASK );

XIntc\_EnableIntr(XPAR\_XPS\_TIMER\_0\_BASEADDR, XPAR\_XPS\_TIMER\_0\_INTERRUPT\_MASK);

XTmrCtr\_SetControlStatusReg(XPAR\_XPS\_TIMER\_0\_BASEADDR, 0, XTC\_CSR\_ENABLE\_TMR\_MASK | XTC\_CSR\_ENABLE\_INT\_MASK |

XTC\_CSR\_AUTO\_RELOAD\_MASK | XTC\_CSR\_DOWN\_COUNT\_MASK);

microblaze\_enable\_interrupts();

InitializeGame(x\_cur, y\_cur);status=PVP;

xil\_printf("-- Game Starts! --\r\n");

xil\_printf("\r\nHuman Player's turn!\r\n");

int vga\_input;

vga\_input=(0<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(1<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(2<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(3<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(4<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(5<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(6<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(7<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(8<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(0<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(0<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(1<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(2<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(3<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(4<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(5<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(6<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(7<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(8<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(0<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(1<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(2<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(3<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(4<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(5<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(6<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(7<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(8<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(0<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(1<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(2<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(3<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(4<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(5<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(6<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(7<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(8<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(1<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(2<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(3<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(4<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(5<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(6<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(7<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(8<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(9<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(10<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(11<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(12<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(13<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(14<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(15<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(16<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(17<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(18<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(19<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(20<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(21<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(22<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(23<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(24<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(25<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(26<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(27<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(28<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(29<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

vga\_input=(9<<24)+(30<<16)+(1<<8)+3;

VGA\_IP\_mWriteReg(XPAR\_VGA\_IP\_0\_BASEADDR, 0, vga\_input);

while (1)

{

if (turn==HUMAN\_PLAYER || (turn==COMPUTER\_PLAYER && status==PVP)) {

do {

if (turn==COMPUTER\_PLAYER && status==PVC)

break;

dip\_check=XGpio\_DiscreteRead(&dip, 1);

StatusReg = XPs2\_GetStatus(&Ps2Inst);

}while((StatusReg & XPS2\_STATUS\_RX\_FULL) == 0);

BytesReceived = XPs2\_Recv(&Ps2Inst, &RxBuffer, 1);

key\_count=(key\_count+1)%3;

if (key\_count==0) {

if (RxBuffer==0x21&& win\_flag==0) {

DrawNumber(level,6,18,EMPTY);

if(level==1)

level=2;

else if(level==2)

level=3;

else

level=1;

if(status==PVC )

DrawNumber(level,6,18,0);

else if(status==CVP)

DrawNumber(level,6,18,1);

else

DrawNumber(level,6,18,EMPTY);

}

if (RxBuffer==0x1D && win\_flag==0) {

EraseCursor(x\_cur, y\_cur);

if (y\_cur<=0)

y\_cur=14;

else

y\_cur--;

DrawChess(x\_cur, y\_cur, CURSOR);

}

if (RxBuffer==0x1B && win\_flag==0) {

EraseCursor(x\_cur, y\_cur);

if (y\_cur>=14)

y\_cur=0;

else

y\_cur++;

DrawChess(x\_cur, y\_cur, CURSOR);

}

if (RxBuffer==0x1C && win\_flag==0) {

EraseCursor(x\_cur, y\_cur);

if (x\_cur<=0)

x\_cur=14;

else

x\_cur--;

DrawChess(x\_cur, y\_cur, CURSOR);

}

if (RxBuffer==0x23 && win\_flag==0) {

EraseCursor(x\_cur, y\_cur);

if (x\_cur>=14)

x\_cur=0;

else

x\_cur++;

DrawChess(x\_cur, y\_cur, CURSOR);

}

if (RxBuffer==0x5A && win\_flag==0) {

DrawBack(3,9,EMPTY);

if (board\_state[x\_cur][y\_cur]==EMPTY) {

if(status==CVP)

DrawChess(x\_cur, y\_cur, 1-turn);

else

DrawChess(x\_cur, y\_cur, turn);

board\_state[x\_cur][y\_cur]=turn;

board\_record[BackTimes].x=x\_cur;

board\_record[BackTimes].y=y\_cur;

BackTimes++;

count=0;

time0=0;

if (turn==COMPUTER\_PLAYER)

step\_flag=1;

if (CheckWin(x\_cur,y\_cur,turn)==1) {

xil\_printf("\r\nHuman Player wins!\r\n");

win\_flag=1;

DrawWinning(0, 1, EMPTY);

if(status==CVP)

DrawWinning(0, 1, 1-turn);

else

DrawWinning(0, 1, turn);

}

if (CheckBan(x\_cur,y\_cur,turn)==1){

xil\_printf("\r\nComputer Player wins!\r\n");

win\_flag=1;

DrawWinning(0, 1, EMPTY);

if(status==CVP)

DrawWinning(0, 1, turn);

else

DrawWinning(0, 1, 1-turn);

}

else {

if (turn==HUMAN\_PLAYER)

turn=COMPUTER\_PLAYER;

else

turn=HUMAN\_PLAYER;

xil\_printf("\r\nComputer Player's turn!\r\n");

}

}

}

if (RxBuffer==0x29 && turn==HUMAN\_PLAYER && win\_flag==0) {

count=0;time0=0;

if (status==PVP) {

x\_cur=7;

y\_cur=7;

for (i=0; i<256; i++) {

board\_record[i].x=0;

board\_record[i].y=0;

}

InitializeGame(x\_cur, y\_cur);status=PVP;

DrawStatus(0, 18, EMPTY,status);

status=PVC;

DrawNumber(level,6,18,0);

DrawStatus(0, 18, COMPUTER\_PLAYER,status);

}

else if(status==PVC) {

x\_cur=7;

y\_cur=7;

for (i=0; i<256; i++) {

board\_record[i].x=0;

board\_record[i].y=0;

}

InitializeGame(x\_cur, y\_cur);status=PVP;

DrawStatus(0, 18, EMPTY,status);

status=CVP;

DrawNumber(level,6,18,1);

DrawStatus(0, 18, COMPUTER\_PLAYER,status);

turn=COMPUTER\_PLAYER;

}

else if(status==CVP) {

x\_cur=7;

y\_cur=7;

for (i=0; i<256; i++) {

board\_record[i].x=0;

board\_record[i].y=0;

}

InitializeGame(x\_cur, y\_cur);status=PVP;

DrawStatus(0, 18, EMPTY,status);

status=PVP;

DrawStatus(0, 18, COMPUTER\_PLAYER,status);

}

}

if (RxBuffer==0x76) {

x\_cur=7;

y\_cur=7;

for (i=0; i<256; i++) {

board\_record[i].x=0;

board\_record[i].y=0;

}

InitializeGame(x\_cur, y\_cur);status=PVP;

}

if (RxBuffer==0x2D) {

if(BackTimes>0){

BackTimes--;

x\_cur=board\_record[BackTimes].x;

y\_cur=board\_record[BackTimes].y;

board\_state[x\_cur][y\_cur]=EMPTY;

DrawChess(x\_cur,y\_cur,EMPTY);

turn=1-turn;

if(status==PVC)

{

BackTimes--;

x\_cur=board\_record[BackTimes].x;

y\_cur=board\_record[BackTimes].y;

board\_state[x\_cur][y\_cur]=EMPTY;

DrawChess(x\_cur,y\_cur,EMPTY);

turn=HUMAN\_PLAYER;

}

DrawBack(3,9,turn);

}

}

}

}

if (turn==COMPUTER\_PLAYER && (status==PVC ||status==CVP )&& win\_flag==0) {

if (step\_flag==0) {

if (x\_cur-1<0)

x\_pos=x\_cur+1;

else

x\_pos=x\_cur-1;

y\_pos=y\_cur;

step\_flag=1;

}

else {

if(level==2||level==3){

EvaluateComputerMove(board\_state, 0, MIN\_INFINITY, MAX\_INFINITY, 0, 0);

x\_pos=maxMoveX;

y\_pos=maxMoveY;

xil\_printf("\r\n computer \r\n");}

else

{

everyScore(Computer);

current\_pos=best(Computer);

x\_pos=current\_pos.y;

y\_pos=current\_pos.x;

xil\_printf("\r\n computer \r\n");

}

}

xil\_printf("\r\n%x, %x\r\n", x\_pos, y\_pos);

if(status==CVP)

DrawChess(x\_pos, y\_pos, 1-turn);

else

DrawChess(x\_pos, y\_pos, turn);

board\_state[x\_pos][y\_pos]=COMPUTER\_PLAYER;

board\_record[BackTimes].x=x\_pos;

board\_record[BackTimes].y=y\_pos;

BackTimes++;

count=0;

time0=0;

if (CheckWin(x\_pos, y\_pos, turn)) {

xil\_printf("\r\nComputer Player wins!\r\n");

win\_flag=1;

DrawWinning(0,1, EMPTY);

if(status==CVP)

DrawWinning(0,1, 1-turn);

else

DrawWinning(0,1, turn);

turn=HUMAN\_PLAYER;

}

else {

turn=HUMAN\_PLAYER;

xil\_printf("\r\nHuman Player's turn!\r\n");

}

}

}

return 0;

}

POINT best(int who) //找出得分最好的点，对于human是最大的，对于computer找最小

{

POINT bestPoint;

int bestValue;

int i,j;

POINT bestPoints[226];

int num=0; //最大值的个数

srand( (unsigned)time( NULL ) ); //初始化随机数种子

if(who==Human) {

bestValue=-99999999;

for(i=0;i<N;i++) {

for(j=0;j<N;j++){

if(pointValue[i][j]>bestValue && board\_state[i][j]==Nothing){

num=0;

bestValue=pointValue[i][j];

bestPoint.x=j;bestPoint.y=i; //注意i，j与x,y的对应

bestPoints[num++]=bestPoint;

}

else if(pointValue[i][j]==bestValue && board\_state[i][j]==Nothing){

bestPoint.x=j;bestPoint.y=i;

bestPoints[num++]=bestPoint;

}

}

}

}

if(who==Computer) {

bestValue=99999999;

for(i=0;i<N;i++) {

for(j=0;j<N;j++){

if(pointValue[i][j]<bestValue && board\_state[i][j]==Nothing){

num=0;

bestValue=pointValue[i][j];

bestPoint.x=j;bestPoint.y=i; //注意i，j与x,y的对应

bestPoints[num++]=bestPoint;

}

else if(pointValue[i][j]==bestValue && board\_state[i][j]==Nothing){

bestPoint.x=j;bestPoint.y=i;

bestPoints[num++]=bestPoint;

}

}

}

}

bestPoint=bestPoints[rand()%num]; //随机选一个

return bestPoint;

}

void everyScore(int who) //遍历每个点评分 需知道当前状态，该谁下子

{

int i,j;

int vData;

for(i=0;i<15;i++){

for(j=0;j<15;j++){

pointValue[i][j]=0;

}

}

for(i=0;i<15;i++){

for(j=0;j<15;j++){

if(board\_state[i][j]==Nothing){

board\_state[i][j]=who; //如果落子

pointValue[i][j]=sumall(board\_state); // 算分

if(jinshou()){

if(who==Computer && first==Computer) pointValue[i][j]=99999999; //禁手

else if(who==Human&& first==Human) pointValue[i][j]=-99999999; //禁手

}

board\_state[i][j]=Nothing; //恢复

}

}

}

}

int sumall(int data[N][N]) //对当前状态评分，

{

int heng,shu,zhengxie,fanxie; //横，竖，正斜，反斜的总分

int i,j,k;

int num=2\*N-1-8; //斜向路数

int \*state[15];

huo3[Computer]=0;huo3[Human]=0;

huo4[Computer]=0;huo4[Human]=0;

for(i=0;i<N;i++){

state[i]=data[i];

}

int line[N];

heng=0;

for(i=0;i<N;i++){heng+=(sumline(state[i],N,Human)-sumline(state[i],N,Computer));}

shu=0;

for(i=0;i<N;i++){

for(j=0;j<N;j++){

line[j]=data[j][i];

}

shu+=(sumline(line,N,Human)-sumline(line,N,Computer));

}

zhengxie=0; // "/" 从n=4 到n=N-5 共num=21路

for(k=4;k<N;k++){

j=0;

for(i=k;i>=0;i--){

line[j]=data[i][k-i];

j++;

}

zhengxie+=(sumline(line,k+1,Human)-sumline(line,k+1,Computer));

}

for(k=N;k<num+4;k++){

j=0;

for(i=N-1;i>=0;i--){

line[j]=data[i][k-i];

j++;

}

zhengxie+=(sumline(line,2\*N-1-k,Human)-sumline(line,2\*N-1-k,Computer));

}

fanxie=0; // "\" 从n=4 到n=N-5 共num=21路

for(k=4;k<N;k++){

j=0;

for(i=N-1-k;i<N;i++){

line[j]=data[i][j];

j++;

}

fanxie+=(sumline(line,k+1,Human)-sumline(line,k+1,Computer));

}

for(k=N;k<num+4;k++){

j=0;

for(i=0;i<=2\*N-k-2;i++){

line[j]=data[i][i+k-N+1];

j++;

}

fanxie+=(sumline(line,29-k,Human)-sumline(line,29-k,Computer));

}

return heng+shu+zhengxie+fanxie;

}

int sumline(int line[],int length,int who) //行评分函数 输入 某一行， 该行长度 对哪一方进行评分

{

int j,k;

int another; //另一方

int side[2]; //左右两边两个是否被堵住，堵住为1，没堵住为0

int num=0;

int numof0=0; //可能链接的最大个数

int zhong0=0; //中间空格

int score=0; //得分

if(who==Human) another=Computer;

else if(who==Computer) another=Human;

else return -1;

for(j=0;j<length;j++){

if(line[j]!=who)continue;

zhong0=0;

side[0]=0;side[1]=0;

num=1;numof0=1;

if(j==0 || line[j-1]==another)side[0]=1; //左侧被封堵

else {

for(k=j-1;k>=0&&line[k]!=another;k--){

numof0++; //左侧无子

}

}

for(j++;j<length && line[j]==who;j++){

if(line[j]==who) {num++;numof0++;}

}

if(j==length) {side[1]=1;} //右侧被封堵

else if(line[j]==Nothing)

{ side[1]=0;

for(k=j;k<length && line[k]!=another;k++){

numof0++;

if(line[k]==Nothing){

if(zhong0==0)zhong0++;

else break;

}

}

if(j+1!=length&&line[j+1]==who){num++;j++;

if(j+1!=length&&line[j+1]==who){num++;j++;

if(j+1!=length&&line[j+1]==who){num++;j++;

if(j+1!=length&&line[j+1]==who){num++;j++;

if(j+1!=length&&line[j+1]==who){num++;j++;

if(j+1!=length&&line[j+1]==who){num++;j++;

}}

}

}

}

}

else zhong0=0;

}

else if(line[j]==another) side[1]=1; //右侧被封堵

score+= p\_Score(num,numof0,side,who,zhong0);

}

return score;

}

int p\_Score(int num,int numof0, int bp[],int who,int zhong0) //算权重分

{

int max = 0;

if(numof0<5){return 0;}

if (num == 5 && zhong0==0)

{

return 1000000; //成5

}

else if (num==5 && zhong0!=0){max+=3000;huo4[who]++;}

else if(num>5 && zhong0==0){

if(first==who)

{ changlian=1;

return 0;} //禁手

else return 1000000;

}

else if (num == 4)

{

if (bp[1] == 1 && bp[0] == 1) // 两边都被堵

{

max += 0;

}

else if (bp[1] == 0 && bp[0] == 0) //活四

{

max += 3000;

huo4[who]++;

}

else

{

max += 900; //冲四

huo4[who]++;

}

}

else if (num == 3)

{

if (bp[0] == 0 && bp[1] == 0)

{

max += 460; //活三

huo3[who]++;

}

else if (bp[0] == 1 && bp[1] == 1)

{

max += 0; //不成五

}

else

{

max += 30; //死三

}

}

else if (num == 2)

{

if (bp[0] == 0 && bp[1] == 0)

{

max += 45; //活二

}

else if (bp[0] == 1 && bp[1] == 1)

{

max += 0;

}

else //死二

{

max += 5;

}

}

else if (num == 1)

{

if (bp[0] == 0 && bp[1] == 0)

{

max += 3;

}

else if (bp[0] == 1 && bp[1] == 1)

{

max += 0;

}

else

{

max += 1;

}

}

return max;

}

/\*

POINT bestboyi(int who)

{

POINT bestPoint;

int bestValue;

int i,j;

POINT bestPoints[226];

int num=0; //最大值的个数

srand( (unsigned)time(NULL)); //初始化随机数种子

if(who==Human) {

bestValue=-99999999;

for(i=0;i<N;i++) {

for(j=0;j<N;j++){

if(pointValue2[i][j]>bestValue && board\_state[i][j]==Nothing){

num=0;

bestValue=pointValue2[i][j];

bestPoint.x=j;bestPoint.y=i; //注意i，j与x,y的对应

bestPoints[num++]=bestPoint;

}

else if(pointValue2[i][j]==bestValue && board\_state[i][j]==Nothing){

bestPoint.x=j;bestPoint.y=i;

bestPoints[num++]=bestPoint;

}

}

}

}

if(who==Computer) {

bestValue=99999999;

for(i=0;i<N;i++) {

for(j=0;j<N;j++){

if(pointValue2[i][j]<bestValue && board\_state[i][j]==Nothing){

num=0;

bestValue=pointValue2[i][j];

bestPoint.x=j;bestPoint.y=i; //注意i，j与x,y的对应

bestPoints[num++]=bestPoint;

}

else if(pointValue2[i][j]==bestValue && board\_state[i][j]==Nothing){

bestPoint.x=j;bestPoint.y=i;

bestPoints[num++]=bestPoint;

}

}

}

}

bestPoint=bestPoints[rand()%num]; //随机选一个

return bestPoint;

}

\*/

int jinshou() //判定禁手 根据first来判定

{

prehuo3[Computer]=huo3[Computer];prehuo3[Human]=huo3[Human];

prehuo4[Computer]=huo4[Computer];prehuo4[Human]=huo4[Human];

sumall(board\_state);

if (huo3[first]-prehuo3[first]>=2) return 1;

if (huo4[first]-prehuo4[first]>=2) return 1;

if(changlian==1) return 1;

return 0;

}