

# 基于 FPGA 的机器博弈五子棋游戏

## 第一部分 设计概述/Design Introduction

### 1.1 设计目的

人工智能是近年来很活跃的研究领域之一，计算机博弈是人工智能研究的重要分支，人工智能中大多以棋类游戏(如象棋、围棋、五子棋等)为例来研究计算机博弈规律，例如 Google 旗下的 AlphaGo 就是选择利用围棋。本设计制作了一个基于 FPGA 的蓝牙笔遥控机器博弈五子棋游戏。

传统的计算机博弈算法一般是使用 CPU 来实现，在博弈算法发展迅速的今天，博弈算法产生的大数据计算效率和棋局评估的准确度是博弈类游戏的两个重要影响因子。由于采用串行计算的 CPU 在运算速度上存在先天的不足，因此本设计使用 FPGA 作为算法的载体，充分发挥 FPGA 并行计算所具有的先天优势，实现了对计算过程的加速从而提高计算效率和准确度。

五子棋是一种深受大众喜爱的游戏，其规则简单,变化多端,非常富有趣味性和消遣性。棋类游戏在具备娱乐性、益智性的同时也因为其载体大多是手机，电脑等移动互联网设备导致现代社会低头族等现象更加严重，危害青少年的身体健康（见下图 1）。同时，移动互联网设备受限于 I/O 设备的数量（如鼠标等），无法实现双人同屏在线游戏，丧失了游戏的一部分趣味性和体验性。因此我们在体感游戏的启发下，设计了可以远程遥控的蓝牙笔和对应的云端平台，给用户提供双人同屏、双人异地等优秀的游戏体验。

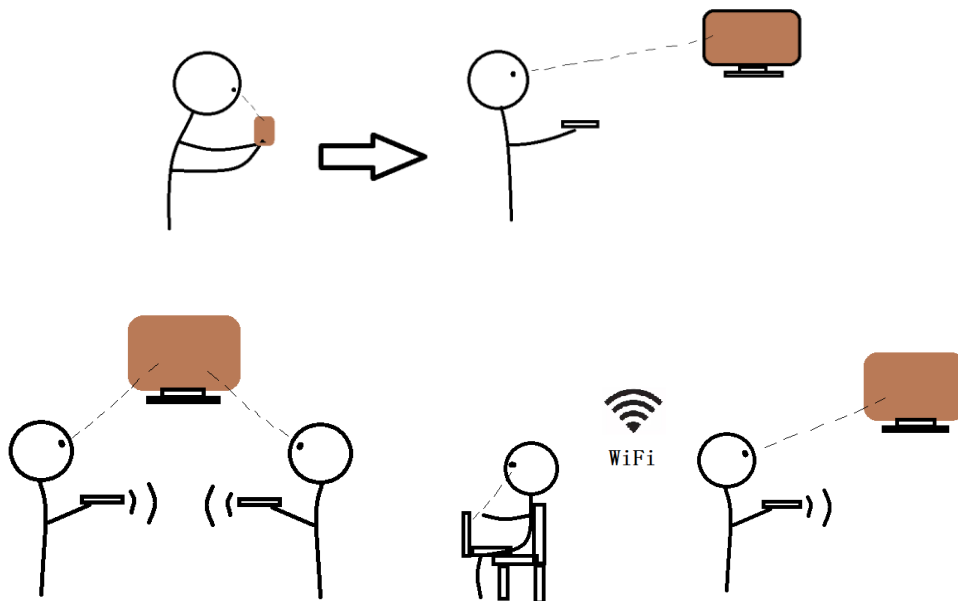


图 1 游戏体验优化图

## 1.2 应用领域

该系统应用十分广泛，既可以作为家庭娱乐工具，也可以作为大数据时代机器学习算法的载体，同时还可以加入更多更有趣的游戏成为体感游戏平台……

## 1.3 适用范围

适用于幼儿教育，家庭娱乐等休闲娱乐领域

## 第二部分 系统组成及功能说明/System Construction & Function Description

### 2.1 系统介绍

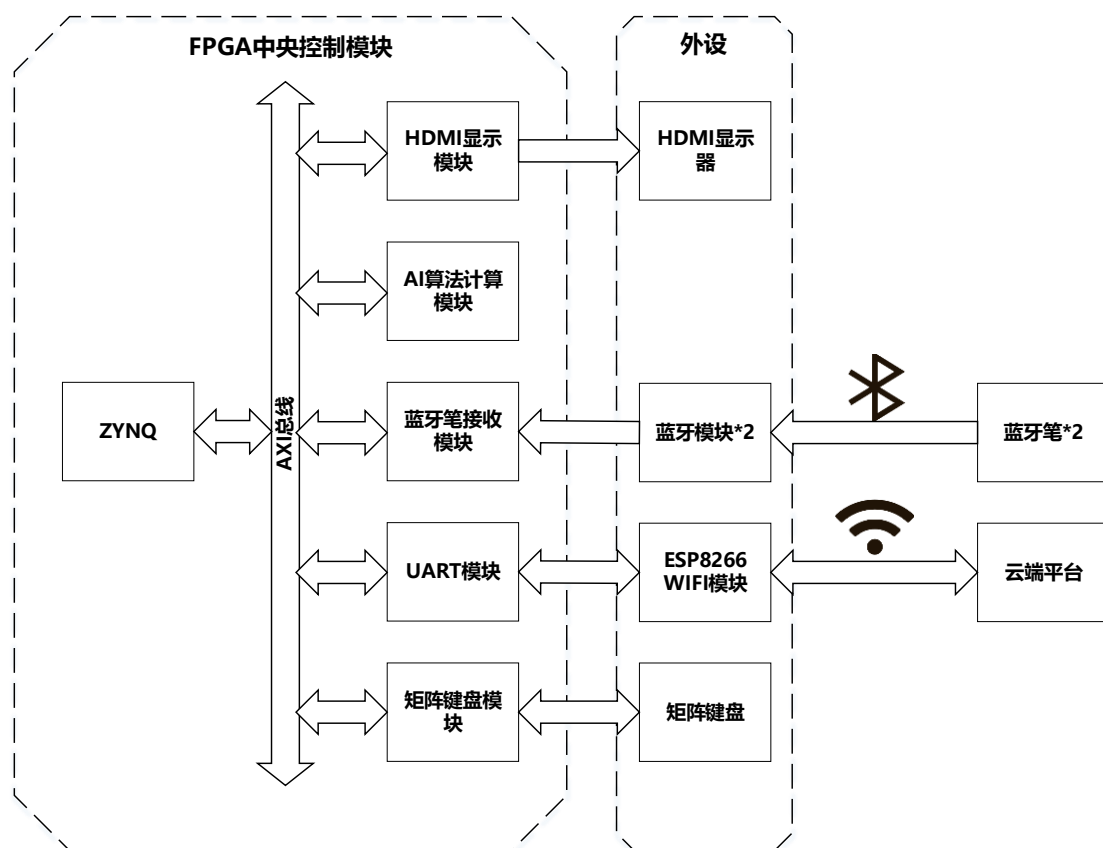


图 2 系统总体结构图

本系统主要由 FPGA 中央控制模块、蓝牙笔模块和云端平台组成，玩家可以通过蓝牙笔或云端平台进行游戏操作，同时游戏的画面会实时显示在 HDMI 显示器上和云端平台。

FPGA 中央控制模块主要负责加速计算五子棋的 AI 算法、对五子棋游戏的流程控制、接收蓝牙笔传来的数据、控制 HDMI 显示，通过 WIFI 发送数据给云端平台，从而实现

## 2.2 各模块介绍

### 2.2.1 FPGA 中央控制模块

#### (1) ZYNQ 控制部分

五子棋的流程我们 ZYNQ 核来控制。五子棋游戏有三种模式：人机对战、人人对战、AI 对 AI(主要是用来训练 AI)，通过玩家手中的蓝牙笔可以进行切换，选择好模式后玩家点击蓝牙笔确认游戏开始，如果是人机对战，玩家可以选择自己执黑棋还是白棋(执黑棋的一方先行)，玩家通过手中的蓝牙笔来控制屏幕上的光标指向，按下确认键落子，然后五子棋 AI 程序进行它的落子，双方轮流进行游戏，直至一方获胜，屏幕上会显示出获胜方，同时指示出五子连线。如果选择了人人对战的模式，那么两个人类玩家可以分别通过两只蓝牙笔轮流下棋，直至一方获胜，屏幕上给出提示。在人机对战和人人对战中，当人类玩家觉得行棋困难时，可以通过蓝牙笔点击提示按钮，AI 程序会计算一个最佳位置显示在屏幕上来提示玩家。在这两种模式下，玩家如果走错棋，都可以通过蓝牙笔点击悔棋按钮悔棋。

具体的游戏流程如下图所示：

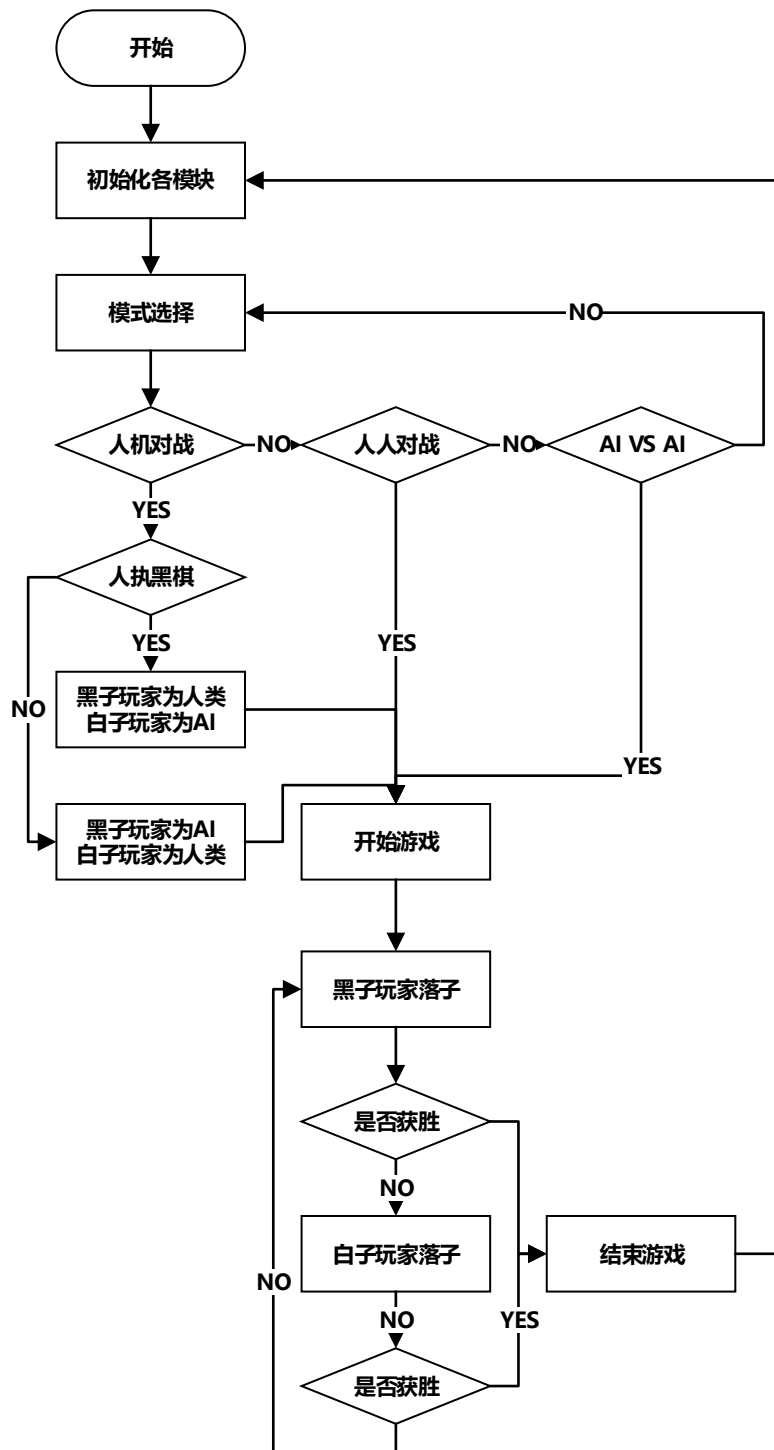


图 3 五子棋游戏流程图

## (2) 五子棋 AI 算法部分

五子棋的 AI 算法我们使用了贪婪算法，对棋盘上每一个未下子的位置进行评

分，选择分值最大的位置作为落子的位置。

根据人类下棋的经验，我们设计了一个评分表，对棋盘上 16 种模型评分：

(1)	“OOOOO” → 50000	(2)	“+OOOO+” → 4320
(3)	“+OOO++” → 720	(4)	“++OOO+” → 720
(5)	“+OO+O+” → 720	(6)	“+O+OO+” → 720
(7)	“OOOO+” → 720	(8)	“+OOOO” → 720
(9)	“OO+OO” → 720	(10)	“O+OOO” → 720
(11)	“OOO+O” → 720	(12)	“++OO++” → 120
(13)	“++O+O+” → 120	(14)	“+O+O++” → 120
(15)	“+++O++” → 20	(16)	“++O+++” → 20

图 4 棋型评分表

普通的五子棋算法使用软件语言实现，进行棋型的模板匹配需要消耗大量时间，并且棋型越多，消耗的时间越长。我们改进了这种算法，使用 HDL 硬件描述语言来实现这一算法，将扫描的一行数据同时进行 16 种棋型的模板匹配，大大加速了棋型匹配算法，提高了我们系统的速度。

具体的实现方法如下图所示：(○ 表示己方棋子，+ 表示空)

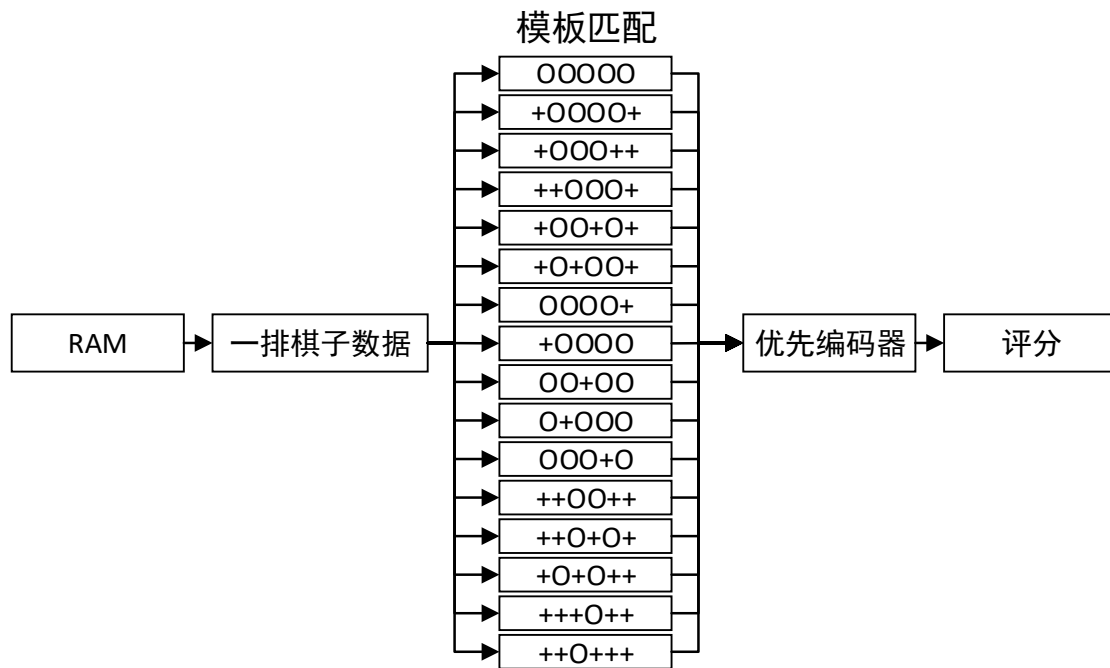


图 5 模板匹配方法

将每个空白落点所在行、列、斜向四个方向的棋子数据从 ram 中取出，然后送入上图的结构进行棋型匹配，将四个方向的得分累加，得到该位置的最终得分。

扫描棋盘上所有空白的位置，找出得分最高的位置作为落子位置。这些位置中可能会有一些位置评分相同，为了保证游戏效果，我们设计了一个真随机数发生器，来确保不会每次 AI 算法都走出相同的走法。真随机数发生器的结构如下图所示：

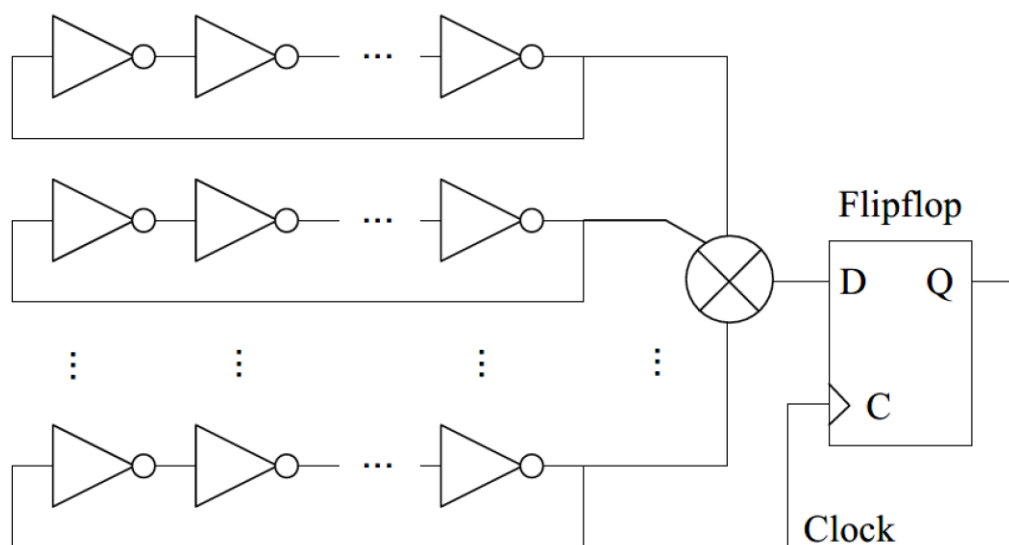


图 6 随机信号发生器

我们将奇数个反相器首位相接，使其自激振荡，产生真实的随机信号。

### (3) HDMI 显示控制部分

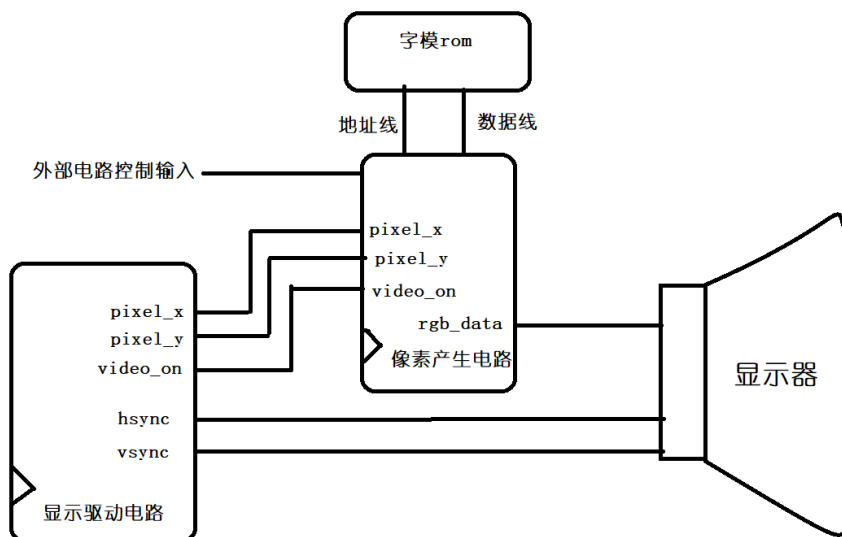


图 7 显示控制部分架构图

如上图 3，由显示驱动电路产生 640\*480 的行、场信号以及扫描点的坐标，送入像素产生电路，像素产生电路会根据点的坐标，判断当前点的区域例



如棋盘或字符显示区域，从而给出响应的 RGB 数据。在字符显示部分中，像素产生电路给字模模块发送一个地址，字模模块返回一行数据，包括开始页面的标题及选项，游戏页面的棋盘行列数、回合数、悔棋等按键、以及光标，当蓝牙笔光标移动到按键部分时，按键自动变色提示玩家。为了节省空间资源，我们采用在同一套子模的基础上采用不同的算法的方式显示大小不同的字符。

#### (1) 显示控制部分：

本系统通过 Verilog 语言硬件描述语言设计出一个 VGA 显示控制器，VGA 显示器因为其输出信息量大，输出形式多样等特点已经成为现在大多数设计的常用输出设备，同时 FPGA 以其结构优势可以使用很少的资源产生 VGA 的控制信号。目前大多数计算机与外部显示设备之间都是通过模拟 VGA 接口连接，计算机内部以数字方式生成的显示图像信息被显卡中的 D/A 转换器转变为 R、G、B 三原色信号和行、场同步信号，信号通过电缆传输到显示设备中（VGA 接口见下图 4）。要实现 VGA 显示就要解决数据来源、数据存储、时序实现等问题，其中关键还是如何实现 VGA 时序。行时序和帧时序都需要产生同步脉冲(Sync a)、显示后沿 (Back porch b)、显示时序段(Display interval c) 和显示前沿(Front porch d)四个部分。本项目采用的是 640X480 (60Hz) 显示屏。

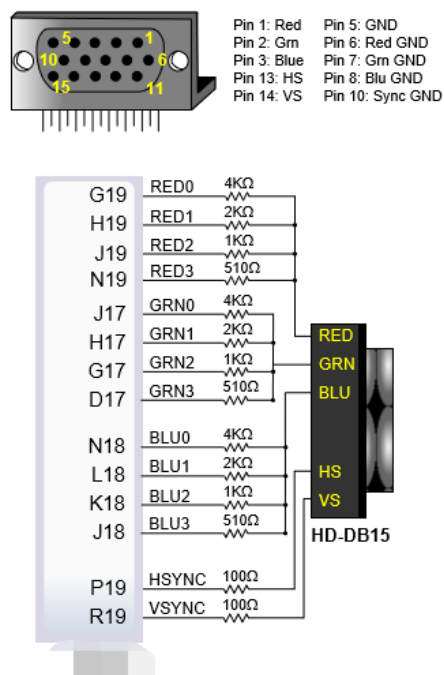


图 8VGA 接口结构图

① VGA 同步电路模块：此模块产生相关时序和同步信号。其核心为一个计数器，计数器的输出信号为 pixel\_x 和 pixel\_y，反映了当前扫描像素的位置，而水平同步信号 hsync 和垂直同步信号 vsynr 也是由内部计数器译码产生，它们与 VGA 端口连接并控制显示器的水平和垂直扫描。VGA 同步电路还产生 video\_on 信号，控制是否使能显示。(VGA 显示控制框图见下图 5)

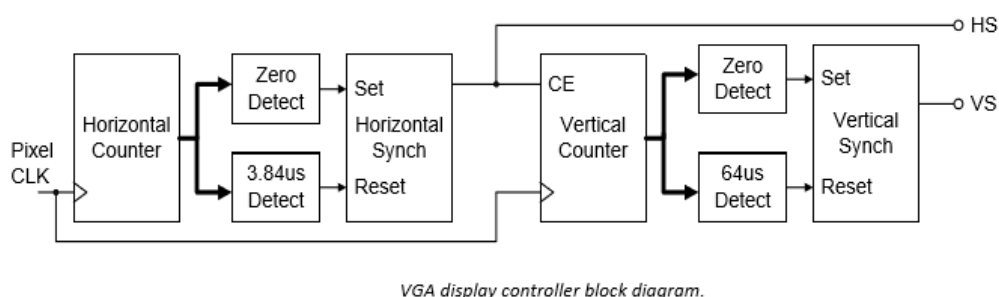


图 9 VGA 显示控制器框图

② 像素产生电路模块：此模块主要产生 RGB 视频信号。

i. 字符显示部分，是使用 FPGA 内部的一个 rom 来存储的字符字库信

息，然后由程序将其提取出来作为显示信号发送到 VGA 接口，以实现字幕和数字的显示，为了降低对存储器的要求，此处采用分区域显示的方法。即将一组像素点构成的显示区域作为显示单元，这里定义一个 8X16 的矩形区域作为显示单元。像素产生电路会根据点的坐标，判断是否在棋盘区域或者字符显示区域，给出相应的 RGB 颜色数据，在显示字符时，像素产生电路给字模模块发送一个地址，字模模块返回一行数据，在确定需要显示的字符后，只需用 pixel\_y[3:0]16 位作为行选，pixel\_x[2:0]8 位用于在一行进行位选即可，字幕中此位信号为 1，屏幕亮字符颜色，为 0 时亮背景颜色，包括开始页面的标题及选项，游戏页面的棋盘行列数、回合数、悔棋等按键、以及光标显示，当蓝牙笔光标移动到按键部分时，按键自动变色提示玩家。通过我们的算法，只使用了一套字模 ROM 就可以显示不同大小的字符，比如在用 8\*16 的字模生成 16\*32 的字符图像时，只要将行选和位选向左移一位，即改为 pixel\_y[4:1]和 pixel\_x[3:1]之后即可，32\*64 像素的图像同理，此算法不易出错又易修改，极大的节省了空间资源。

ii. 图片显示部分，本系统采用的 RGB 为 RGB444，即红色 4 位、绿色 4 位、蓝色 4 位来进行表示。将图片内容存储在 FPGA 的内部的自定义 IP 核中，然后通过 VGA 控制模块，送入坐标，依次得到 ROM 中的值，直接得到此点 RGB 颜色数据，赋值给 RGB\_data 输出引脚。

iii. (3) AXI 自定义 IP 核。显示部分全部完成后打包成 AXI IP 核，通过读取 IP 核中 slv\_reg 的值，实时完成各种显示变化，比如回合数等，此 slv\_reg 可在 MicroBlaze 中进行修改赋值，方便与云端同步。

### 2.2.2 蓝牙笔模块

### (1) 三轴加速度传感器模块

三轴加速度传感器模块由 mpu9150 和 stm32 芯片构成，用于生成 x 轴、y 轴的位移量和按键信息。将两块芯片集成到一个模块上，大大减小了整个模块的体积。

三轴加速度传感器模块使用 UART 串口进行数据传输，具体传输的数据格式如下：

8bits/hex	8bits/hex	8bits/hex	8bits/hex	8bits/hex	8bits/hex	8bits/hex	8bits/hex
AA	AA	07	03	button	X	Y	SUM

(AHRS 模块上没有按键，需要外接按键，如下图所示)，X 为鼠标左右移动量，Y 为鼠标上下移动量。SUM 为 button+X+Y，波特率 115200。

button 有 4 种取值

0：无按键按下、1：左键单击、2：左键双击、3：左键长按、4：右键单击

### (2) 蓝牙模块

蓝牙笔通过蓝牙模块和 FPGA 中央控制模块进行通信。

#### 2.2.1 云端平台

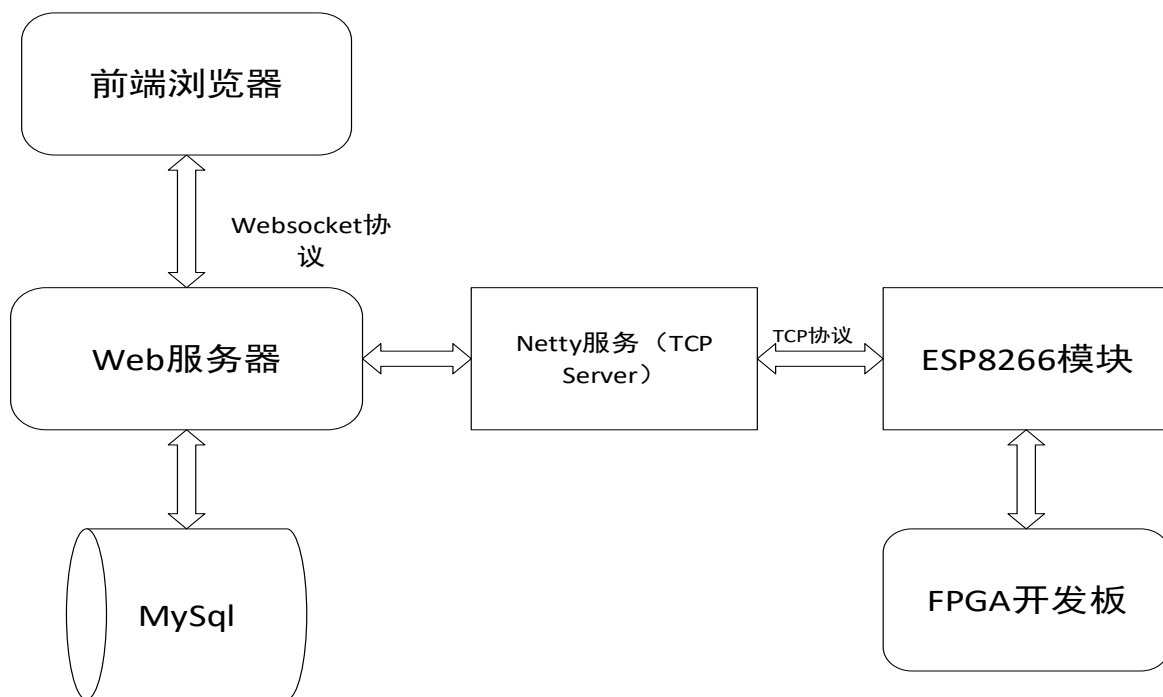


图 10 云端平台架构

云端平台设计架构如上所示,本系统共分为三层：浏览器作为**表现层**；Web服务器以及 netty 服务作为**应用服务层**；Mysql 数据库作为**数据持久层**；FPGA 开发板以及 ESP8266 作为**数据感知层**。

数据感知层中 FPGA 开发板作为数据产生的来源，通过 ESP8266 对数据进行编码并且通过 TCP 协议将数据发送到应用服务层。应用服务层中，使用 netty 网络框架搭建 TCP 服务端接收数据感知层的数据并作出一定的处理。处理后的数据一方面进入数据持久层存入 Mysql 数据库中，另一方面通过 Websocket 协议和页面渲染技术展示在前端表现层。

### 1.数据感知层实现;

为了完成异地、双人的游戏体验，本系统采用 ESP8266 wifi 模块作为数据传输到应用服务层的渠道，通过局域网内 TCP 协议进行数据传输，在 FPGA 开发板和上层应用之间建立双向的数据链接。

ESP8266 模块设置 AT 指令如下图 7:

```
AT+CWJAP="Demo","12345678"//将模块添加至局域网中，引号内为wifi名，密码
AT+CWAUTOCONN=1//开机自启STA
AT+SAVETRANSLINK=1,"192.168.2.211",8088,"TCP"//将TCP连接设置保存到FLASH
AT+CIPMODE=1//开启透传模式，将串口数据直接发送到TCP Sever端
AT+CWDHCP=1,1//开启dhcp
```

图 11 AT 指令图

## 2.应用服务层实现

本系统采用主流网络 NIO 框架 netty 搭建 TCP 服务，避免了多线程拥塞，数据编/解码异常，TCP 半包、粘包等问题。Netty 框架搭建 TCP 服务的流程如下图

8，初始化 bootstrap 具体流程如下图 9

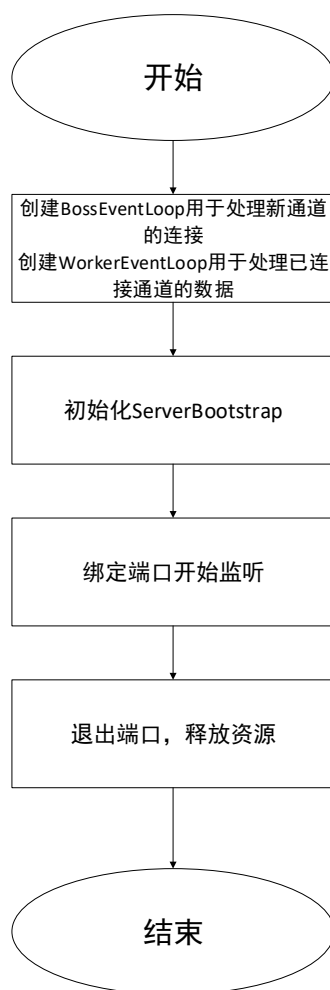


图 12 TCP 服务初始化流程图

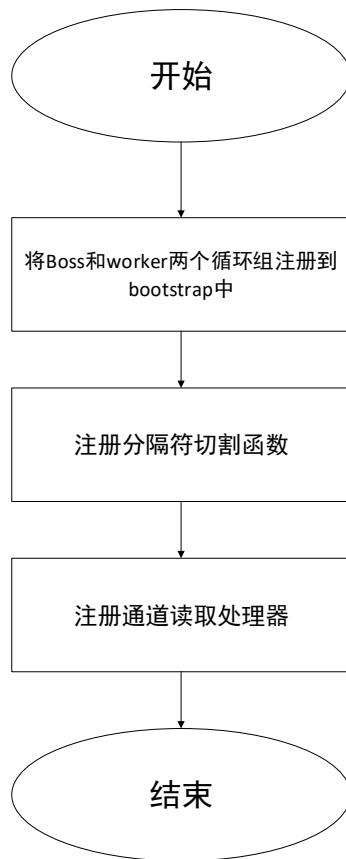


图 12 bootstrap 初始化流程图

数据进入通道后将通过切割函数进行切割避免 TCP 粘包、半包的问题，然后进入读取处理器进行进一步的数据处理。

### 3.数据持久层

选用 mysql 小型关系型数据库对数据进行管理，在简单小巧，可移植性高的前提下可以处理上千万条记录的大型数据。 数据库的表单设计如下图 10，E-R 图如下图 11。



图 13 数据库表单设计图

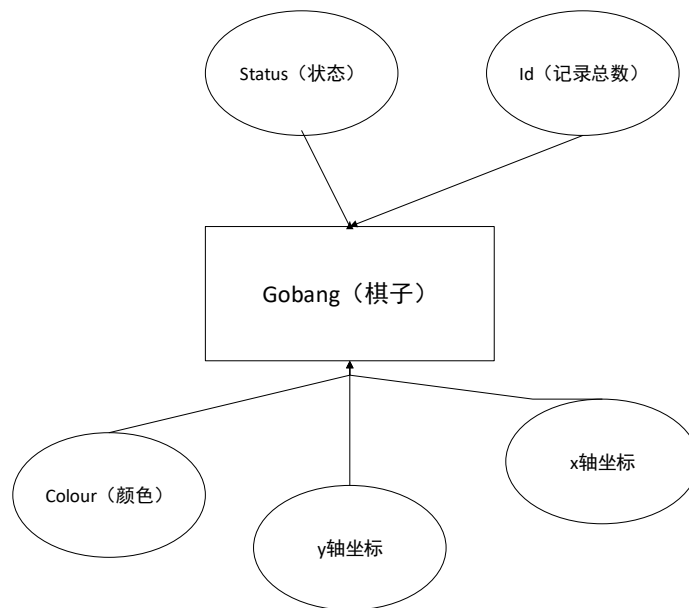


图 14 数据库 E-R 图

id 为总记录数量，随记录的增加自增；status 是当前系统的状态，具体请参考附件 1 中的状态表。Color 是当前棋子的颜色，01 为黑，02 为白；coordinateX/Y 分别代表当前棋子的 x/y 轴坐标的位置。

#### 4. 表现层

表现层采用 websocket 协议与应用服务层进行全双工通信，避免了 Ajax 轮询导致的占用大量资源和丢包的可能性。同时通过 html 和 css 技术渲染出完整的五子棋游戏，实现相关的下子，悔棋，提示，选择模式等功能。

在接受数据感知层数据和发送数据到感知层的过程中，会在收发的过程中导致接受数据的过程触发发送过程，形成死循环。为避免这种情况我们可以采用标记位 flag 进行流程的监督，流程图如下图 12.。



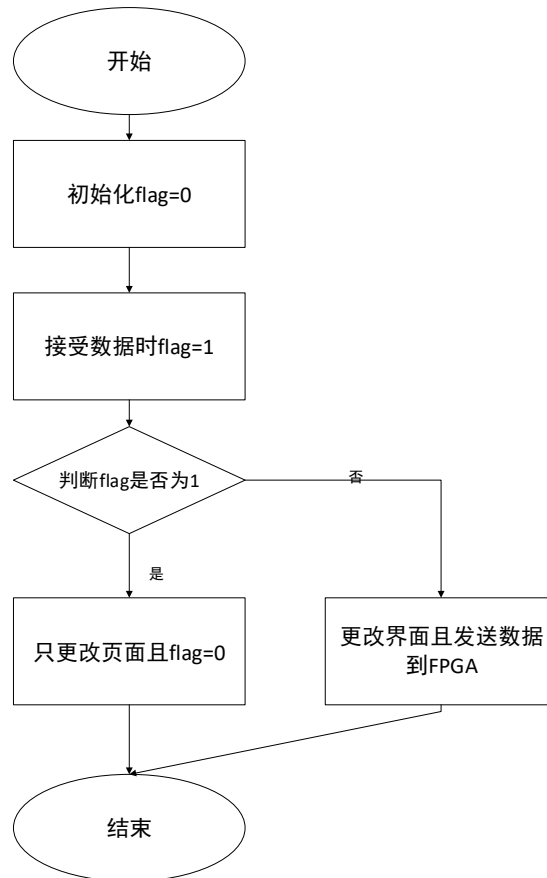


图 15 flag 控制流程图

当从数据感知层接受到数据则将 flag 置 1，在发送数据到感知层时判断是否重复发送，避免在接受数据的同时发送数据形成死循环，并将 flag 置 0 保证下一次的正常数据发送。

### 第三部分 完成情况及性能参数/Final Design & Performance Parameters

#### 3.2.1 FPGA 中央控制模块

##### (1) 显示控制部分：

显示部分分为两个画面：开始页面、游戏页面。

①开始页面见下图 13



图 16 开始页面 VGA 显示

## ②游戏页面

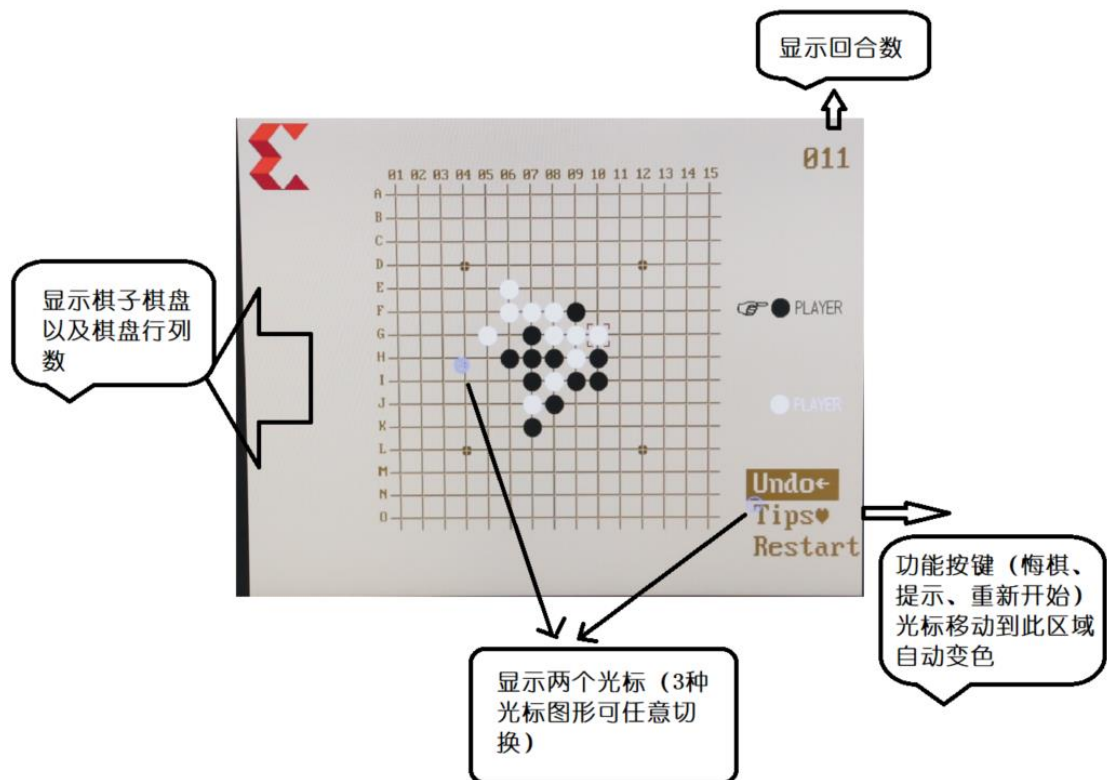


图 17 游戏页面

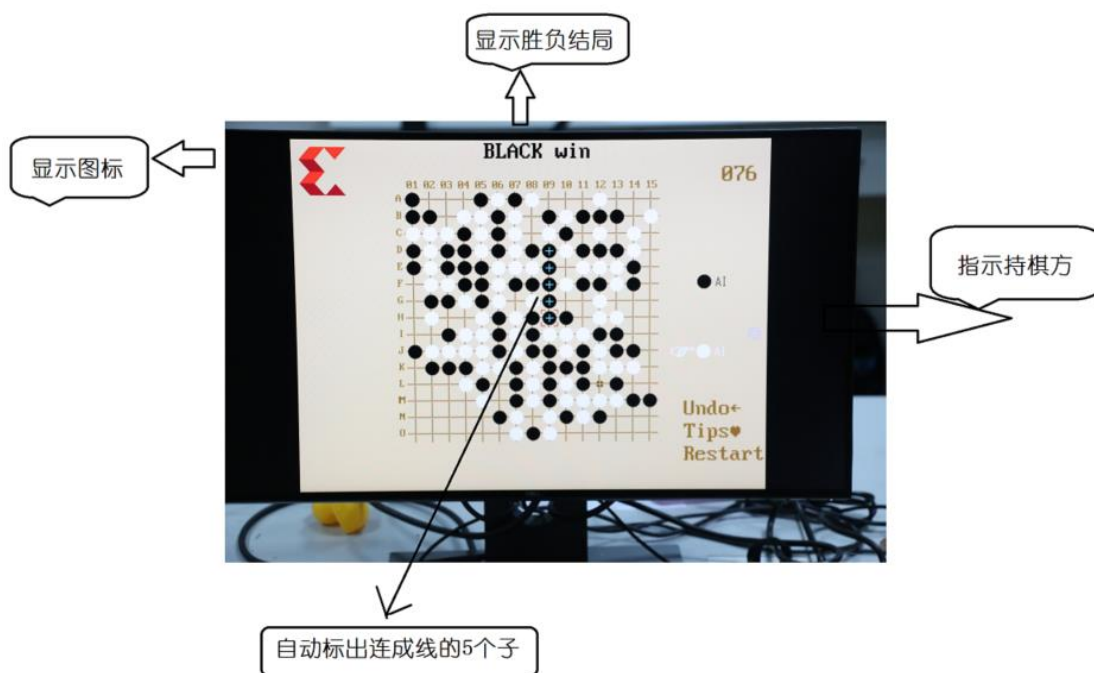


图 18 游戏胜利

## (2) AI 算法加速效果：

传统的 AI 算法使用软件语言实现，实现一层棋盘的模板匹配需要的时间随着棋型的增多会加长，而本算法使用 FPGA 加速后，匹配一个棋盘位置只需要 4 个时钟周期，同时匹配 16 种棋型，棋型匹配所需要的时间大大减少。

## (3) 与云端平台的通信

FPGA 中央控制器与云端平台通信使用 ZYNQ 控制，目前可以流畅地传输数据而不会出现数据丢包的现象。

### 3.2.2 蓝牙笔模块

蓝牙笔模块每 2.5 ms 发送一次位移数据给 FPGA 中央控制器，FPGA 中央控制器将位移量累加，转换为屏幕上的坐标值。因此，蓝牙笔的光标可以流畅的在屏幕上显示出来。按键信号通过中断通知 ZYNQ 控制器，保证了程序运行的流畅性。

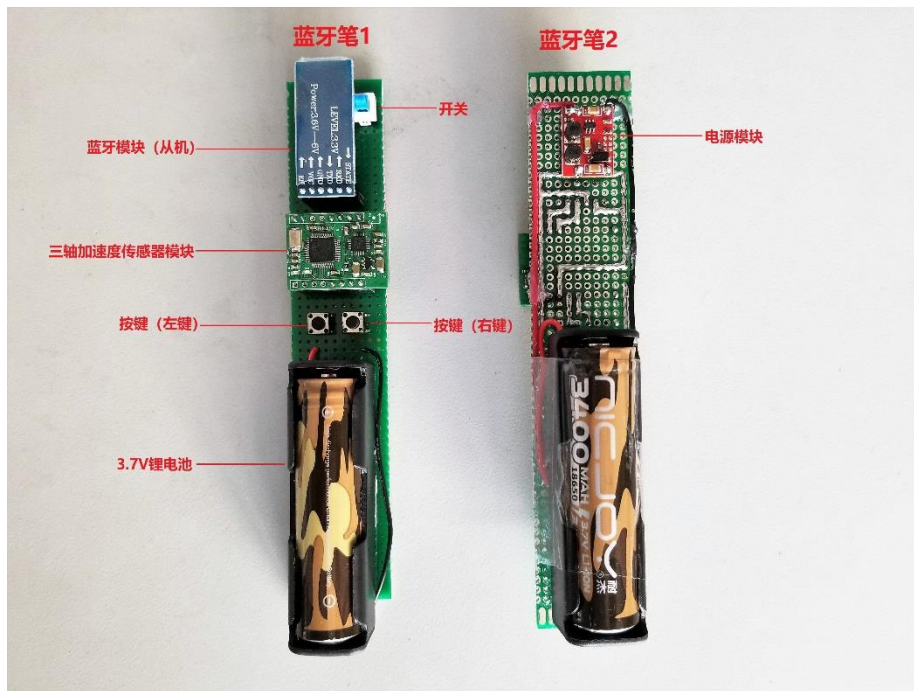


图 19 蓝牙笔结构图

### 3.2.3 云端平台

(1) 应用服务层接受 FPGA 开发板发送的数据，并将数据存入数据库供前端调用。

串口发送数据如下图 16，数据格式意义详情见附录 1 数据标志位表：

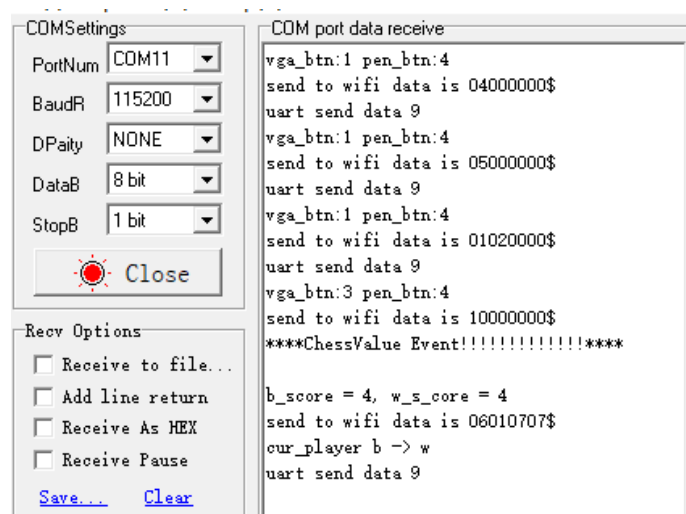
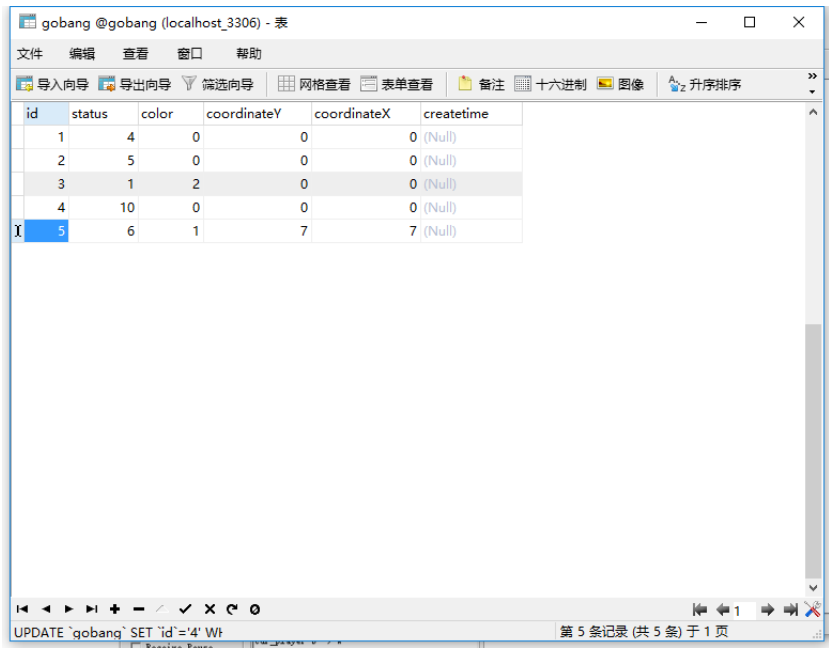


图 20 串口数据图

服务层接受数据后数据库中存入数据如下图 17：



id	status	color	coordinateY	coordinateX	createtime
1	4	0	0	0	0 (Null)
2	5	0	0	0	0 (Null)
3	1	2	0	0	0 (Null)
4	10	0	0	0	0 (Null)
5	6	1	7	7	0 (Null)

图 21 数据库读入数据

(2) 前端对数据进行渲染，实现和 FPGA 开发板相同的游戏功能，如下子，选择模式，悔棋，提示，重新开始等，部分结果如下图 18、19.

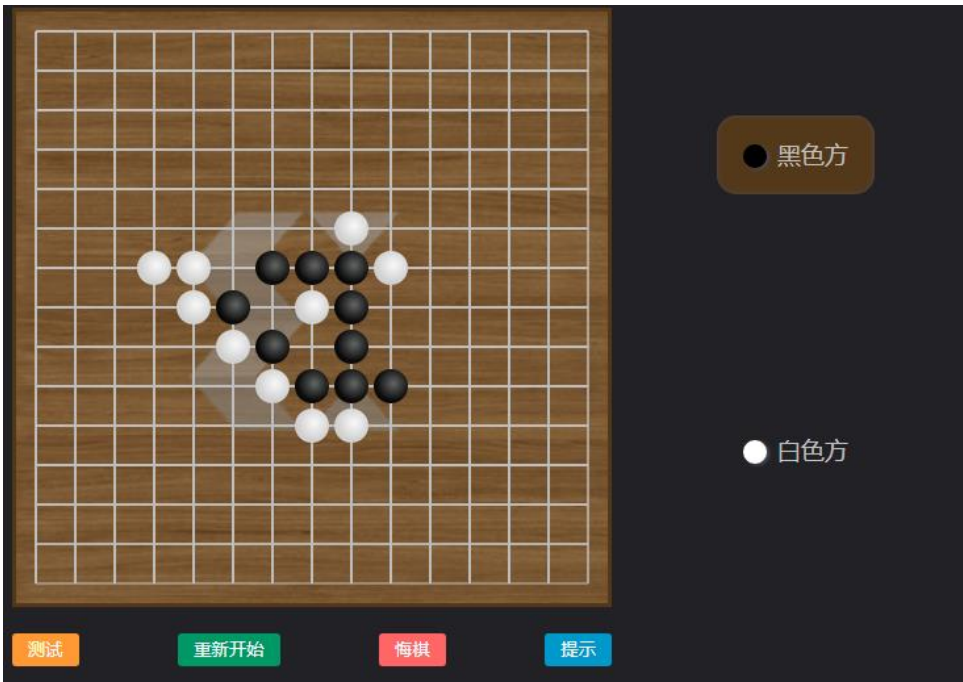


图 22 白子获胜图



图 23 人机对战主页面

### 3.2.4 系统整体结构图

综上，最终本系统整合上述三大模块见下图：

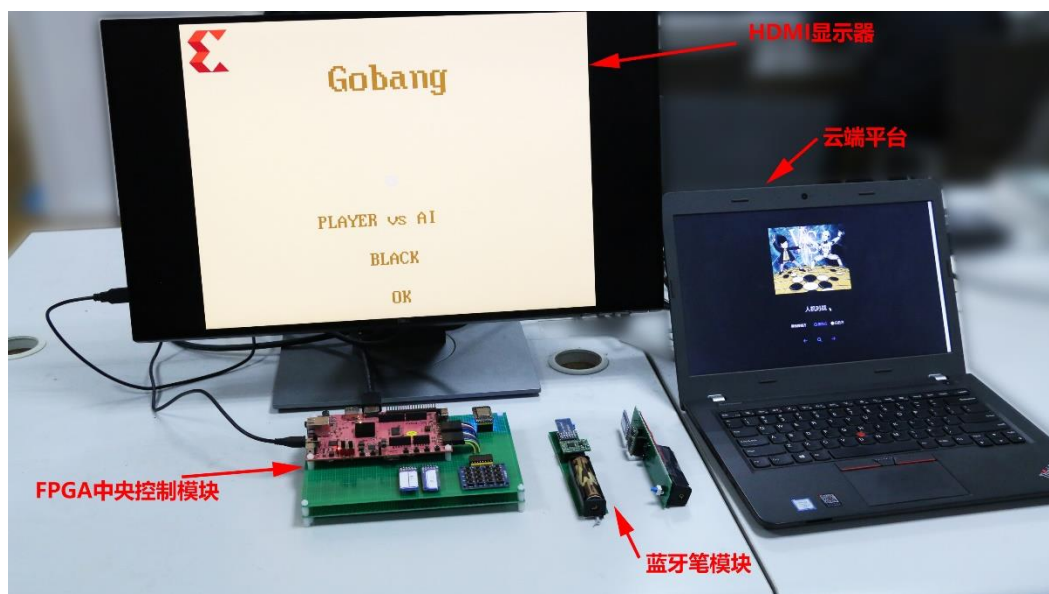


图 24 系统整体架构图

## 第四部分 总结/Conclusions

### 4.1 主要创新点

1 VGA 界面显示不同于传统的五子棋游戏，像素风的游戏风格、多样的鼠标样式、高度可视化的游戏提示，都打破了传统 VGA 游戏的局限性，开创了新型 FPGA 游戏设计的风格。

2.游戏操作由传统的鼠标等 I/O 设备转化为无线蓝牙笔，弥补了双人对战时鼠标无法单独进行操作的缺点，优化了用户的操作体验

3.在传统硬件游戏上添加了同步的云端平台，可以实现双人异地在线游戏，在一定程度上突破了空间对游戏体验的限制。

4，使用 Verilog 重写博弈算法，利用 FPGA 并行计算的先天优势加速了计算过程，提高了计算效率和准确度，优化了游戏体验。

5.云端数据传输时，通过 websocket 技术使后台主动向前端发送数据，避免轮询查询数据导致资源占用过多。

### 4.2 可扩展之处

1.可以在五子棋游戏的基础上搭建 FPGA 体感游戏平台，添加更多更有趣的游戏和设备，同步云端开发，实现设备、游戏、云端三位一体的游戏平台。

2.在 FPGA 上搭建 TensorFlow 平台，训练更复杂、更智能的卷积神经网络算法。



### 4.3 心得体会

1.通过这次设计我们对数字系统设计有了更深刻的理解，认识到了时序、状态机在系统设计中举足轻重的地位，为今后的系统设计打下了铺垫。

2.通过这次设计我们对 FPGA 在并行计算上的先天优势，对 FPGA 和人工智能的关系有了一定的了解

3.同时我们也对 Xilinx 平台有了更充分的了解和应用，Xilinx 的内部资源较丰富，软件支持也较友好。

最后，在此感谢指导老师、组委会老师以及 Xilinx 平台提供的支持和指导！

## 第五部分 附录

附录 1：状态表

1	人机简单	ID_PVA_EASY
2	人机一般	ID_PVA_GENERAL
3	人机困难	ID_PVA_HARD
4	人人	ID_PVP
5	AI VS AI	ID_AVA
6	下棋	ID_LAOZI
7	悔棋	ID_RETRACT
8	获胜	ID_WIN
9	清空棋盘（再来一局）	ID_RESTART
10	开始游戏	ID_START_GAME
11	接收错误	ID_RECV_ERROR
12	落子提示	ID_HINT

附录 2：block design 图，详细代码见附件



