2016/12/6

Tennis For Two 设计说明

数字电路实验大设计

姓名：王若冰　　学号：PB15121735

Tennis For Two 设计说明

数字电路实验大设计

设计摘要

世界上第一个用电子技术实现的电子游戏诞生于布洛克海芬国家实验室(Brookhaven National Laboratories)，运用纯粹电子技术实现的游戏在如今已经并不多见，本设计初衷是为了向世界上第一个电子游戏致敬，因为它几乎彻底改变了人类休闲娱乐的方式。本设计在当初设计的基础上也加入了现代游戏的一些元素，使游戏具有更高的可玩性和互动性。

开发平台

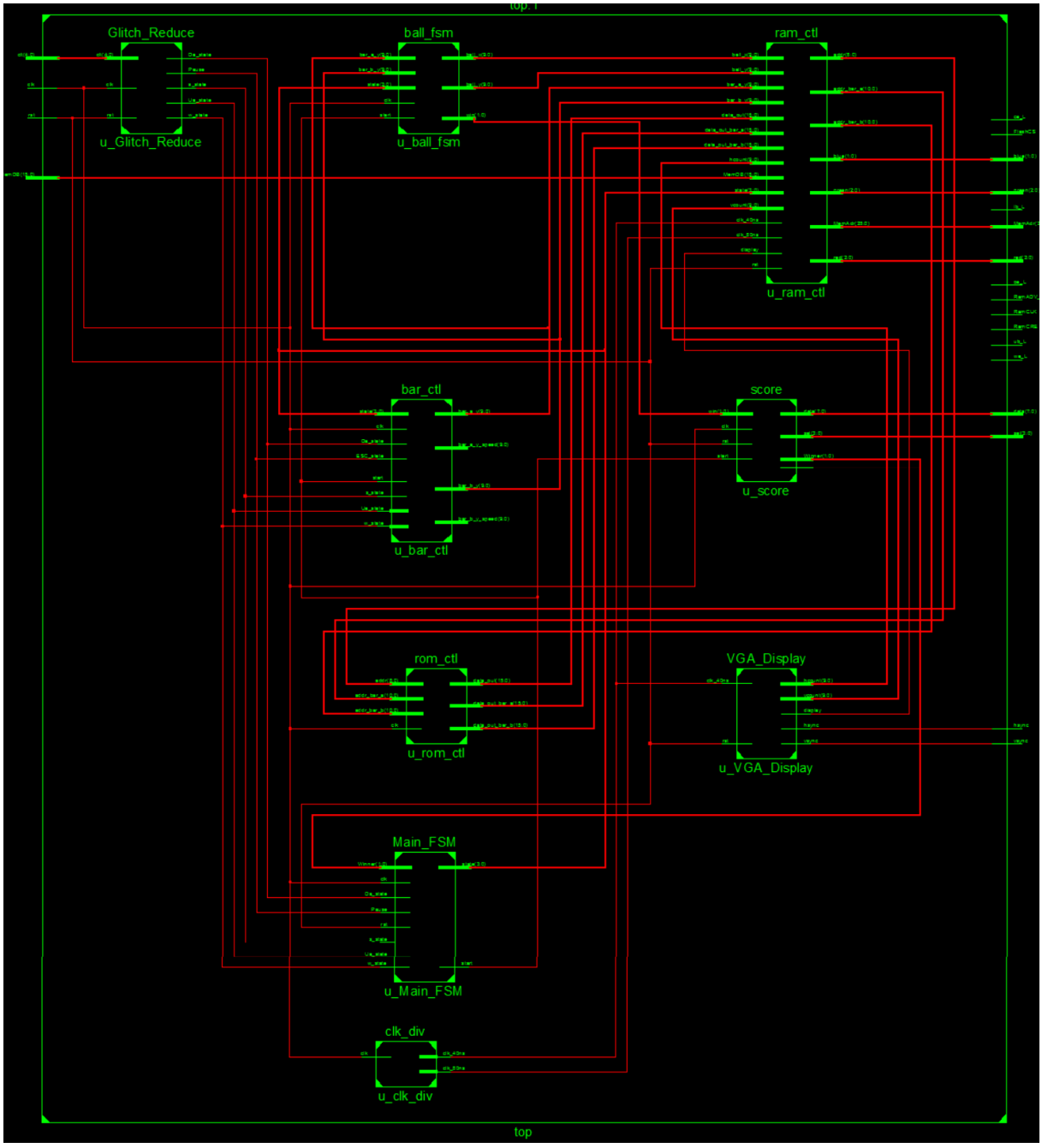
本设计使用的 EDA 工具是 ISE14.7，开发板的型号为 DIGILENT 公司的 Nexys 3，FPGA 型号为 Spartan-6 XC6SLX16CSG324C。

实验外设

1. VGA 屏幕。
2. 开发板上的 16Mbyte Micro Cellular RAM。
3. 开发板上的 5 个按钮以及 1 个开关。
4. 开发板上的数码管。

设计框架

总设计电路连接图



The RTL Schematic of Top Level

如上图所示，本设计一共使用了 9 个模块。每个模块之间的功能相对独立，使代码在编辑会更加逻辑清晰。现在分别将各个模块的功能阐述如下。由于源代码有将近 2000 行，所以具体代码可以直接参看附件中的工程文件，不在赘述。

Glitch_Reduce 模块

该模块的功能是将按钮的信号进行去抖动处理，由于按钮信号将会作用于主状态机。而一次按下按钮，会产生较多次抖动，影响状态机的正常工作，所以这一步必不可少。

clk_div 模块

该模块的作用是产生 40ns 以及 80ns 信号的时钟信号。40ns 的信号是用于 VGA 的显示所需，由于该设计用的屏幕分辨率为 640×480，该分辨率所需的信号周期是 40ns。而对于 RAM 的物理设计，有一个重要的时序要求，就是读写周期不能小于 70ns。这意味着，将某一个需要读的地址放入地址端，在 70ns 后结果才会出现在结果输出端口中。这样就要求我们对于读数据和显示数据的时序需要有个巧妙的设计。一个周期为 80ns 的时钟正好满足这样的要求：在每 80ns 中更新一次 RAM 读端口的地址，每一次读出的数据为 16bit，这正好是两个像素的 RGB 值，所以在下一个 80ns 中每 40ns 都会将上次读出的数据分别用于前后两个像素的显示 RGB 值。

Main_FSM 模块

该模块是整个设计的工作核心（但是并不是最复杂技术核心，只是逻辑上的核心），也是整个电路的总状态机。整个设计大的状态有 10 个，其中三个是在总菜单上面状态，一个是在游戏过程中的状态，一个是显示说明的状态，一个是退出状态，还有两个暂停界面状态以及三个结算界面的状态（分别对应显示左边胜利、右边胜利和平局）。状态间的转换是根据按钮控制实现。总的状态机按照时序逻辑电路的分类应该属于 Moore 型时序逻辑电路。

ball_fsm 模块

该模块是整个设计较为复杂的状态控制模块，用于控制球的运动。针对球在碰撞到边界，碰撞到球板的各个不同方向都需要编写不同的分类。在刚开始设计的时候，笔者本来准备设计球拍可以上下左右移动。但是如果这样，这种状态机的状态转换条件就会非常繁多和复杂。在编写了将近 600 行的一个 always@模块之后，实际的运行状态出现了诸多意想不到、难以调试的 bug，最后笔者觉得，将情况化简为只能上下移动的球拍。如此整个球控状态机的复杂度几乎减小了一半，运行结果也十分理想，而且也使得游戏更加容易上手，运用有限的按钮和开关对于游戏的操作也显得更加人性。

故这种简化处理其实是一举两得的，不仅使得设计过程得到简化，从用户的角度来讲，这种简化也更加人性。也就是所谓的，避免了 over-engineering.

bar_ctl 模块

这个模块较为简单，功能主要是用于控制球板的运动，根据设计的游戏地图，球板的运动是有所限制的。由于有上下挡板的限制，球板并不能到达界面的最上和最下，这一点需要在地图设计上将挡板的设计精确到像素。从而就可以使得整个界面的运行看起来十分的自然。

ram_ctl 模块

这个模块可以说是本设计最复杂的一个模块，其作用是通过接受主状态机、球控状态机、球板状态机的控制来驱动对于 ram 或者 rom 的读操作。由于 IP 核的 512KB 的限制，IP 核不能用于存储较大的背景图。所以背景图我选用的是 FPGA 上面的 16Mbyte 的 RAM 来存储。一共有 10 张背景图，所对应的每个背景图都是 306560 Bytes 的大小，所以将十张背景图存储在 RAM 上就占有约 2.92MB 的空间。该模块由于作为了 VGA 信号的 RGB 端口的最后控制模块，也需要根据球和球板的位置发送相关信号去控制 rom 的读图操作。

rom_ctl 模块

这个模块是作为 IP 核的控制驱动。本设计使用了 3 个 IP 核，分别用于存储球的图像信息以及左和右球板的图像信息。图像信息的 coe 文件是笔者用 Mathematica 生成，其具体过程和原理下面会具体叙述。这个控制驱动会根据 ram_ctl 发送过来的信号进行读操作并将最后的结果返回到 ram_ctl 中，在其控制下作为最后的 RGB 信号输出。

score 模块

这个模块的工作是根据球控模块发送的输赢信号进行记分，并将结果显示在数码管上，而且还会作为一个记分器，在用于游戏最后的评判胜负的依据。

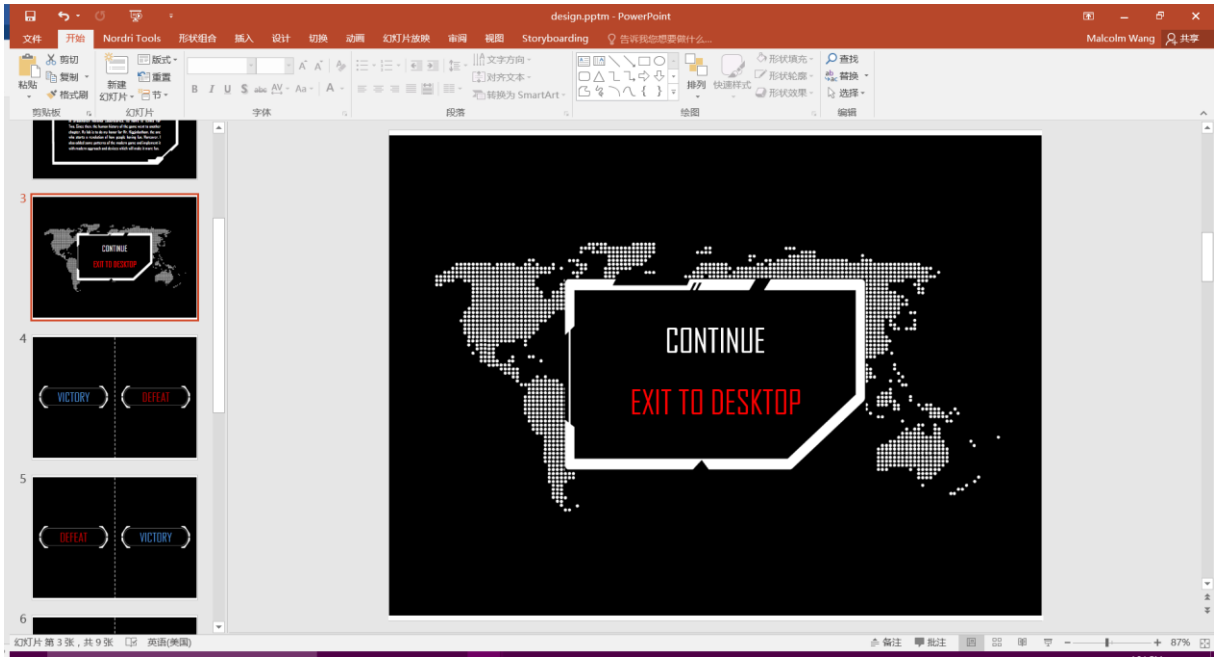
VGA_Display 模块

该模块的作用是用于产生 VGA 信号所需的行同步和场同步信号，以及生成具体的显示行数和列数用于 ram 中的控制。

图像界面设计

User Interface 设计

对于 UI 设计，由于在游戏背景中需要精确到像素，所以最简单的实现方法是使用 PowerPoint 进行设计。由于 PPT 中可以精确调整各个图片的位置，所以设计图像设计过程得到了一定的化简。具体的图形界面设计 PPT 草稿也位于附件中，如下选取了其中一个暂停界面的 PPT 设计截图示意。



Design of User Interface on PowerPoint

技术实现

设计好后的 PPT 画面可以方便的转化为图片格式，但是最重要的是将图片格式转化为在 FPGA 内存上储存的二进制信息。其实网上存在很多这样的软件，但是难以满足本设计的具体要求。由于本设计既需要生成纯粹的二进制文件来初始化 16MB 的 RAM 内存，也需要生成 coe 文件来初始 IP 核的内存。而且前者需要将 10 张图片的数据先后写入一个文件下载到内存中，所以需要根据自己的设计要求进行图像处理应用的编写。笔者在这里使用的 Mathematica 进行图片的二进制化处理。

根据要求，我们需要将每一个点的像素的 RGB 值转换为数字信号。根据 Nexys3 的 VGA 信号端口设计，其实只能输出数字信号而非模拟信号。因此，该端口输出的信号的极限是显示 256 种颜色。所以笔者使用 8bit 表示每个像素的 RGB 值，分别是红色用三位，绿色用 3 位，蓝色用 2 位。应该需要将图片中的信号近似化，近似的函数如下的 Mathematica 代码表示。

```
FitToNexys3RG[x_] := Piecewise[{{0, x < 1/14}, {1, 1/14 ≤ x < 3/14}, {2, 3/14 ≤ x < 5/14}, {3, 5/14 ≤ x < 7/14}, {4, 7/14 ≤ x < 9/14}, {5, 9/14 ≤ x < 11/14},
    {6, 11/14 ≤ x < 13/14}, {7, 13/14 ≤ x <= 14/14}}];
FitToNexys3B[x_] := Piecewise[{{0, x < 1/6}, {1, 1/6 < x ≤ 3/6}, {2, 3/6 < x ≤ 5/6}, {3, 5/6 ≤ 1}}];
converter[{x_, y_, z_}] := FitToNexys3RG[x] * 32 + FitToNexys3RG[y] * 4 + FitToNexys3B[z];
```

Mathematica Program for Color Approximation

如上所示，其实是用一个分段函数，依据四舍五入的原则将颜色信号来使颜色信号在最后的显示中最小的失真。然而经过笔者的测试，最好还是在图像设计中多使用标准色，最后的显示结果才会更加理想。

RAM 上初始化文件格式较为简单，就是使用的纯二进制进行烧写。所以在 RAM 上初始化文件的生成使用的代码如下：

```
In[157]:= FitToNexys3RG[x_] := Piecewise[{{0, x < 1/14}, {1, 1/14 ≤ x < 3/14}, {2, 3/14 ≤ x < 5/14}, {3, 5/14 ≤ x < 7/14}, {4, 7/14 ≤ x < 9/14}, {5, 9/14 ≤ x < 11/14},
        {6, 11/14 ≤ x < 13/14}, {7, 13/14 ≤ x <= 14/14}}];
    FitToNexys3B[x_] := Piecewise[{{0, x < 1/6}, {1, 1/6 < x ≤ 3/6}, {2, 3/6 < x ≤ 5/6}, {3, 5/6 ≤ 1}}];
    converter[{x_, y_, z_}] := FitToNexys3RG[x] * 32 + FitToNexys3RG[y] * 4 + FitToNexys3B[z];
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Interface1.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Interface2.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Interface3.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\background.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\AboutMe.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Pause1.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Pause2.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Victory1.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Victory2.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    f = Import["D:\\Files built by Me\\ISE\\pic_game\\Victory3.bmp"];
    BinaryWrite["D:\\Files built by Me\\ISE\\pic_game\\Background_bin", Thread[converter[Flatten[ImageData[f], 1]]]]
    Close["D:\\Files built by Me\\ISE\\pic_game\\Background_bin"]
```

Mathematica Program for Generating RAM Initialization File

Mathematica 代码也在附件中，文件名为 “pic_process_v3.0.nb”。在其他电脑中运行上述代码的时候需要注意的是要把图片目录根据具体情况进行修改。

与此同时我们需要对 IP 核上的内存进行初始化，这个需要用同样的原理生成 coe 文件。而 coe 文件其实是字符编写的文件，而不是二进制编写的文件。生成过程有所不同，其 Mathematica 代码如下：

```
FitToNexys3RGcoe[x_] := Piecewise[{{"000", x < 1/14}, {"001", 1/14 ≤ x < 3/14}, {"010", 3/14 ≤ x < 5/14}, {"011", 5/14 ≤ x < 7/14}, {"100", 7/14 ≤ x < 9/14},
    {"101", 9/14 ≤ x < 11/14}, {"110", 11/14 ≤ x < 13/14}, {"111", 13/14 ≤ x <= 14/14}}];
FitToNexys3Bcoe[x_] := Piecewise[{{"00", x < 1/6}, {"01", 1/6 < x ≤ 3/6}, {"10", 3/6 < x ≤ 5/6}, {"11", 5/6 ≤ 1}}];
f = Import["D:\\Files built by Me\\ISE\\pic_game\\ball.bmp"];
data1 = Flatten[ImageData[f], 1];
Export["D:\\Files built by Me\\ISE\\pic_game\\ball_coe.txt",
  Flatten[{"memory_initialization_radix=2;\nmemory_initialization_vector=",
    Flatten[Flatten[Table[If[i = Length[data1], List[FitToNexys3RGcoe[data1[[i]][[1]]] <> FitToNexys3RGcoe[data1[[i]][[2]]] <> FitToNexys3Bcoe[data1[[i]][[3]]] <> ";"],
      List[FitToNexys3RGcoe[data1[[i]][[1]]] <> FitToNexys3RGcoe[data1[[i]][[2]]] <> FitToNexys3Bcoe[data1[[i]][[3]]] <> ","]], {i, 1, Length[data1]}]], 1]}];
```

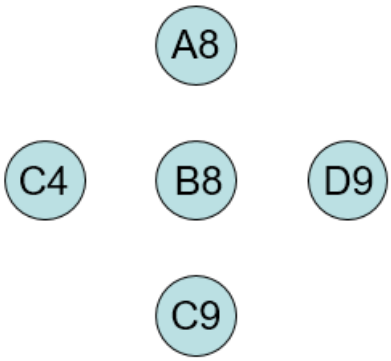
Mathematica Program for Generating COE File

如上生成的是球图像的 coe 文件，在生成后可以方便的先查看编写的情况，然后将文件的拓展名从 txt 改成 coe，就可以进行 IP 核的初始化了。

附件中可以找到所有设计图片，Mathematica 代码以及生成的各种内存初始化文件，方便进行工程的重建。

设计说明

本实验使用的是 Nexys3，主要的操作是按钮，开关仅仅作为一个复位键（状态为 1 的时候复位）。按钮在开发板上的位置如下，在游戏正在进行的时候，C4 和 C9 分别是左边玩家的球板上、下键，而 A8 和 D9 则是右边玩家的上下键，B8 键为暂停键。而在非游戏进行过程时候，即在主界面、暂停界面等时，A8，C9 则是分别上下键，控制选择选项的上下，D9 的作用则类似 Enter 为确定键，C4 则类似于 Esc 为返回键。



Position of Buttons and its Names in FPGA

设计技术点

课内学习的技术点

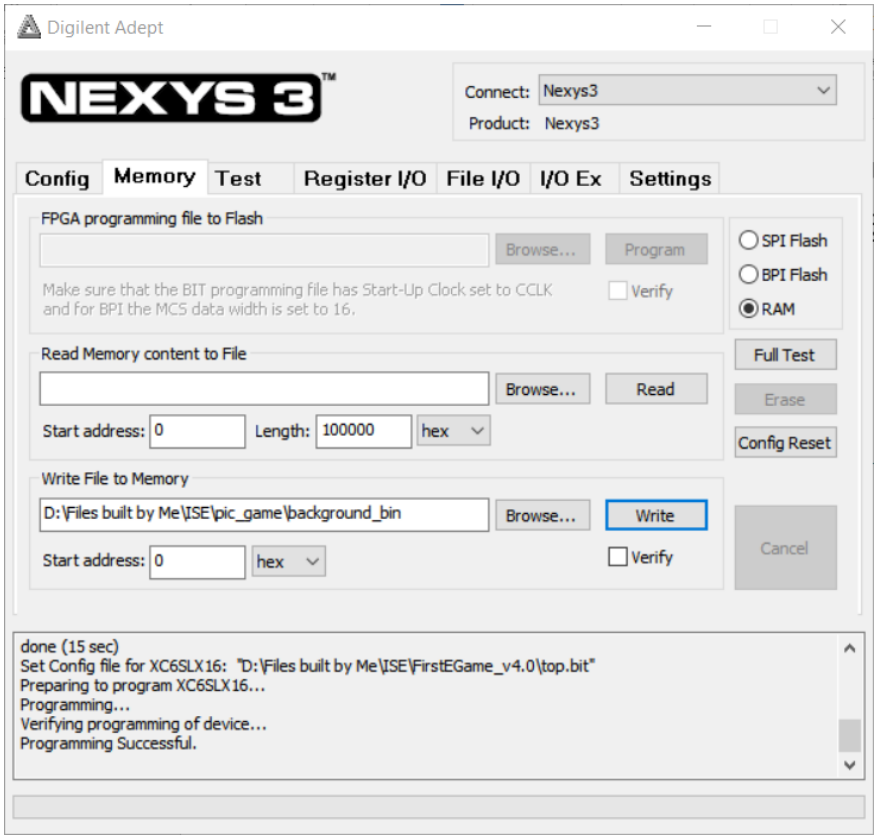
1. 开关去抖动模块的编写。
2. 时钟分频模块的编写。
3. 记分板的使用设计数码管的显示技术。
4. 状态机的设计思想。虽然整个设计其实是非常多的状态机的嵌套和复杂的组合，但是课内通过简单例子掌握的状态机设计方法并没有太大不同。只是在实现的逻辑上会复杂很多。
5. IP 核的使用。本设计使用了 IP 核对于一些小的图片元素进行储存，否则，全依靠 Cellular RAM 进行处理在时间上是不够的。

自学掌握的技术点

1. VGA 的时序及其使用，以及 VGA 驱动的编写。
2. 16MB Micron Cellular RAM 的读写时序以及使用。由于需要储存较大的图片信息，IP 核的存储内存是不够的，所以由于设计需要自学了 Nexys3 中的 RAM 的用法，并根据自己的需求编写了驱动模块。
3. Mathematica 进行图片的二进制化处理。由于需要将图片的每个像素的 RGB 值按照具体的设计要求以两种不同文件形式（二进制文件和 coe 文件）依次存储。所以需要根据自己的要求编写图片处理应用。正好笔者的 Mathematica 曾经有自学，掌握的较为熟练，因此可以借用其强大的内部函数库用较少的代码完成设计要求。

设计演示

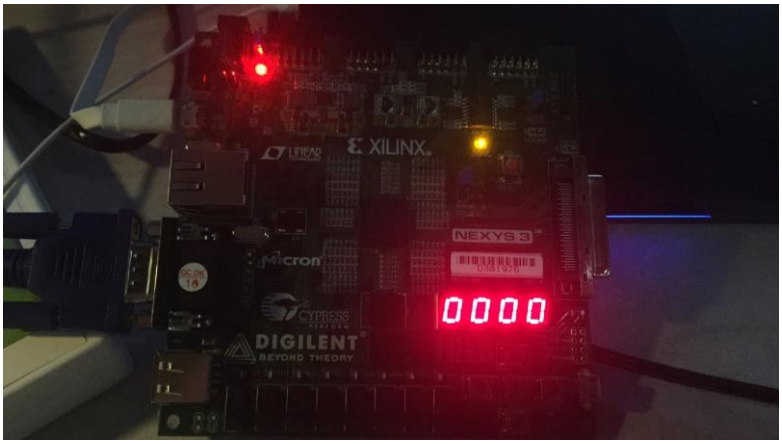
首先，在测试的开始有一个值得一提的点就是与普通的工程文件不同，在下载 bit 文件之前，要先下载内存数据。因为该程序用了开发板上的 RAM，而这个外设的初始化需要单独进行，方法如图所示。注意右边勾选的是 RAM，然后打开前面已经生成好的背景二进制文件，点击 write 即可。



The Configuration of RAM

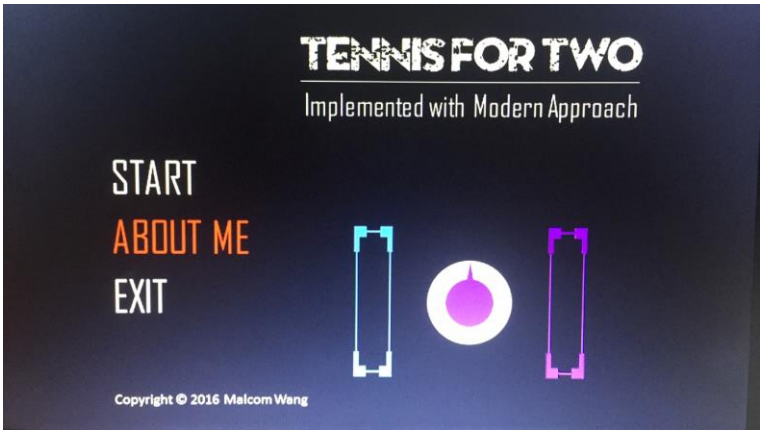
接着下载 bit 文件，完毕后，屏幕和开发板如下图所示。

The Main Menu



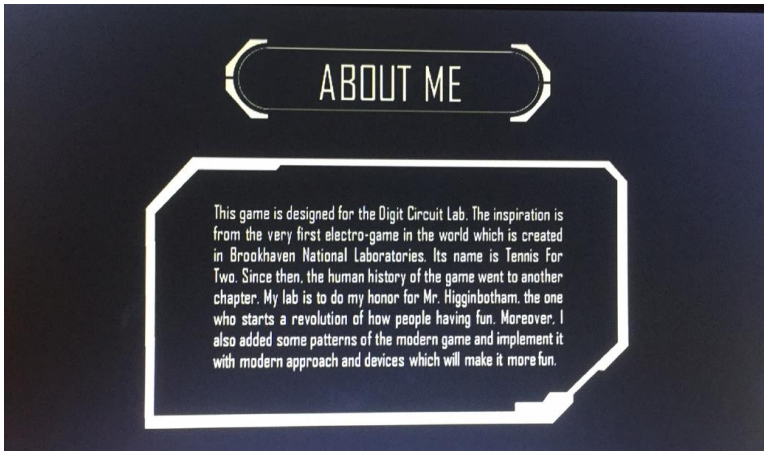
The FPGA at beginning

按上下按钮可以将改变当前红色选定的选项，如下按下键，屏幕显示为：



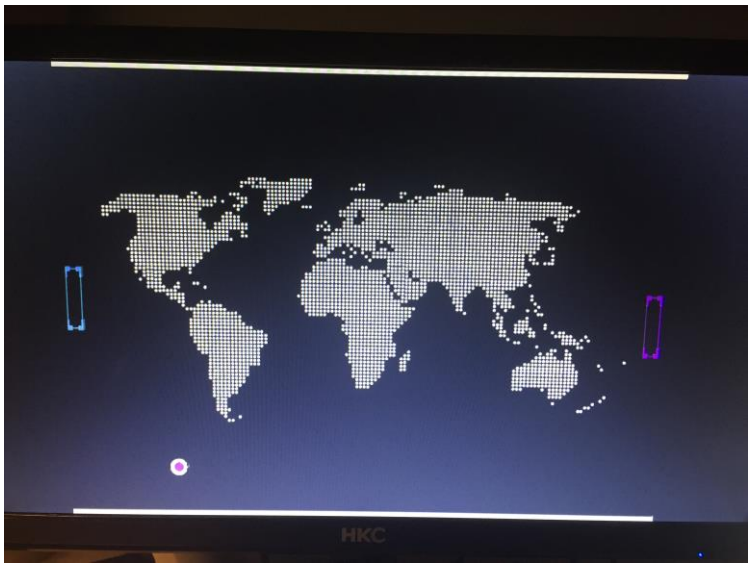
The screen when pressed the downwards button

再按一下右按键则会进入 About Me，屏幕显示为：



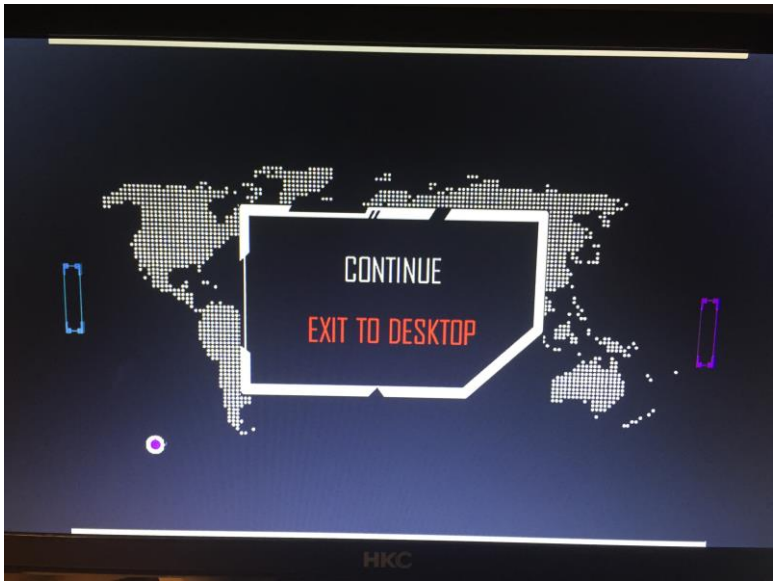
The content of ABOUT ME

按一下左键退出到主菜单后，我们继续选择 START，进入正式游戏。具体的游戏界面如下：



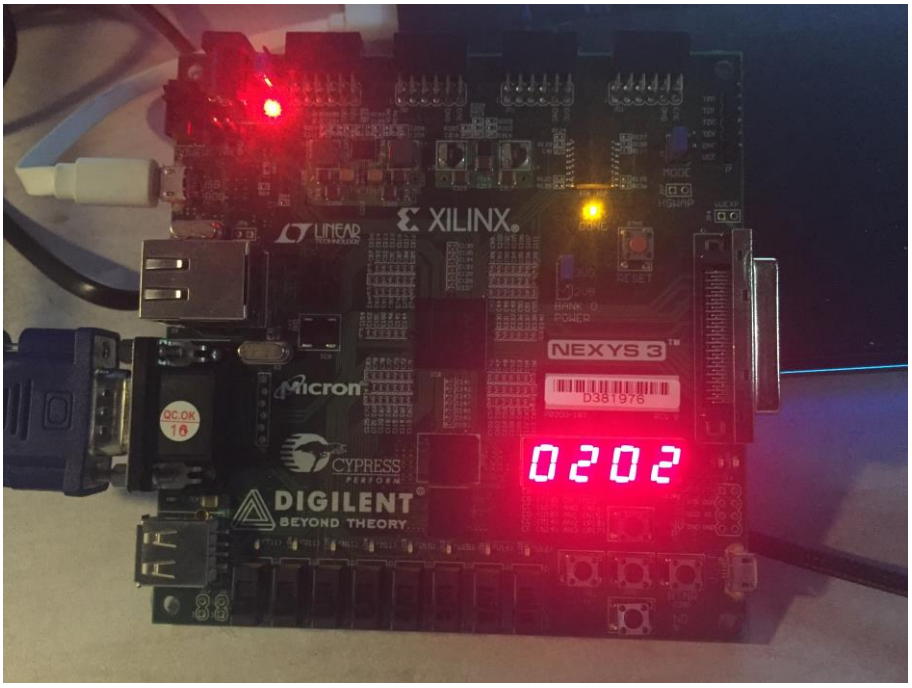
The interface of game

在该界面中的键位操作就如设计说明中叙述的一样，当一方没有接到球的时候，另一方将得一分，并且球将回到初始点。最后球经过一段等待时间后（该时间的目的是让双方玩家做好准备），将重新发出。在游戏的时候均可以按中间按钮来暂停游戏，暂停时游戏界面如下：

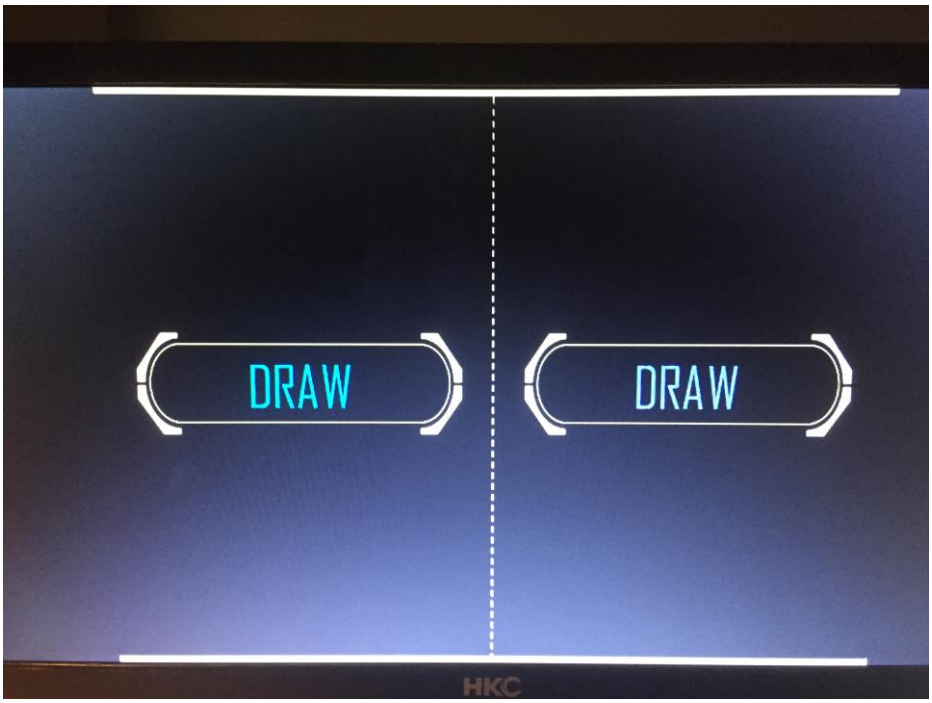


The interface of paused time

若选择了 Exit To Desktop，游戏则先会根据当前数据进行结算，比如当前记分板上的数据为 2:2，则结算界面会显示平局，如图所示。当然根据分数也会有左边胜的结算界面和右边胜的结算界面。



The Score Board



The interface of draw

设计总结

本设计总体来说是比较复杂度、稳定性和完整性上来说都是较为不错的。由于使用了 FPGA 上面的内存，从硬件上支持了较为美观的界面设计，所以设计展示出来的效果比较好。本设计从技术上来说，综合了很多方面的知识，比如 Mathematica 进行图片二进制化处理，用户界面的设计，复杂状态机的编写。在实现的过程中，笔者针对课内没有涉及的两种外设，VGA 和 Cellular RAM 内存的使用进行了自学。最后实现出来的效果也比较理想。

该实验美中不足的是，并没有实现键盘的使用。如果有键盘的存在，键位的操作会更加简便。笔者在设计的过程中也有所考虑，甚至其实在设计的一开始就已经将 PS/2 键盘的驱动端口模块写好，可惜的是笔者所用的机械键盘使用的 USB-HID 键盘协议，与 Nexys3 上面的 USB 端口使用的 PS/2 键盘协议不一致。笔者的键盘不能用于代码的调试，所以放弃了这种想法，转而设计优化按钮的键位设置。虽然也得到了一定的优化，最后的操作效果也并不差，但是还是不如能直接使用键盘来输入的操作更加方便。除此之外，笔者对于游戏性的设计并没有做到最佳，本来设想的是在游戏地图的一些区域加上一些场。如匀加速场，球进入该场的范围会在一个特定方向做匀加速运动，模仿电场或者重力场的效果。以及旋转场，球进入该须臾就会根据其速度过一个圆周运动，模仿静态的磁场的效果。但是无奈时间有限，加入这些场之后，不仅球控状态机需要做一定的修改，游戏的背景也要做一定的调整，所以该游戏设计在最后并没有加入在最终的设计中。这也是本设计的一个遗憾吧。

但是总的来说，笔者对于该设计还是比较满意的，也明显感觉到了在实现这个设计的过程中自己的代码能力的提高非常明显。工程越到后期，代码出错率越小。代码的简洁程度，代码的编写效率在设计后期也和前期有较大的提高。淫才，该实验不仅使笔者学到了很多硬件方面的知识，而且对于笔者自身能力的提高也是有显著效果的。