计算机硬件系统设计

第三次实验报告

Tetris

班　　级：07111503

指导老师：LCY、 WJ

组　　长：11201517XX HSF

小组成员：11201517XX CMQ

小组成员：11201517XX L W

小组成员：11201517XX WZW

学　　院：计算机学院

视频演示： https://youtu.be/vm-R8dQgQTM

目录

一、 实验目的

使用 Altera 公司的 DE2i-150 开发板与硬件描述语言(HDL)设计一个俄罗斯方块(Tetris)游戏机，游戏界面通过 VGA 输出在显示屏上，通过 FPGA 开发板上的按钮控制游戏重置、方块组平行移动和方块组旋转。

二、 环境配置

Table 1 环境配置

软件应用环境	**Quartus II 13.0（64-bit）**
仿真工具	**ModelSim-Altera 10.1d**
硬件开发平台	**DE2i-150**
显示器分辨率	**640*480**
编程语言	**硬件描述语言-Verilog HDL**

三、 设计规划

1. Tetris 项目介绍

Tetris 是一款运行于 DE2i-150 开发板上的，通过 VGA 显示的俄罗斯方块游戏项目。

该项目使用硬件描述语言 Verilog HDL 进行设计，经历工程建立、设计输入、综合与编译、编程下载和修正润色五个阶段。

Tetris 游戏规则如下:

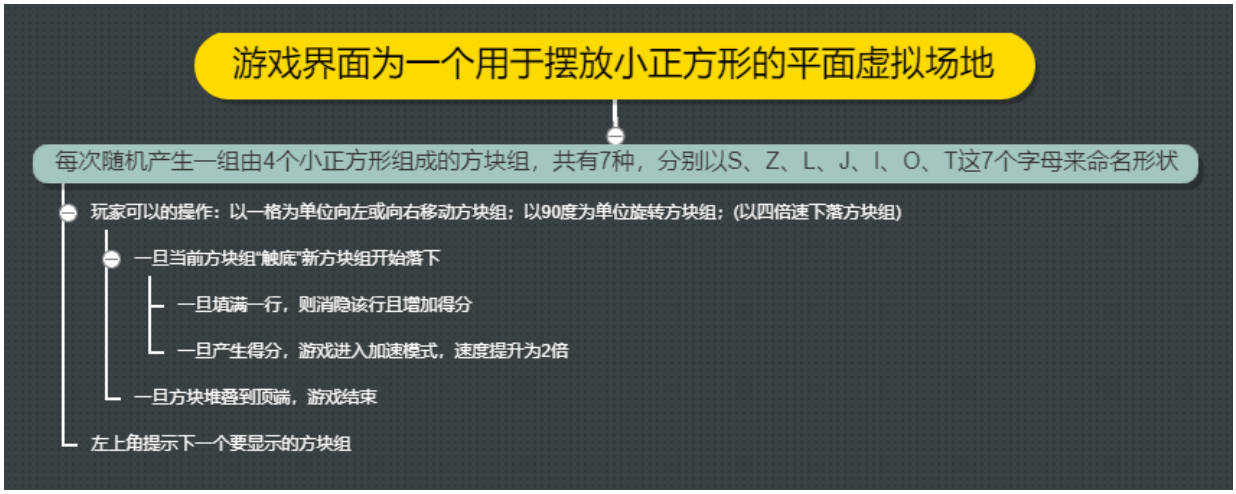

Figure 1 游戏规则

2. 模块与功能设计

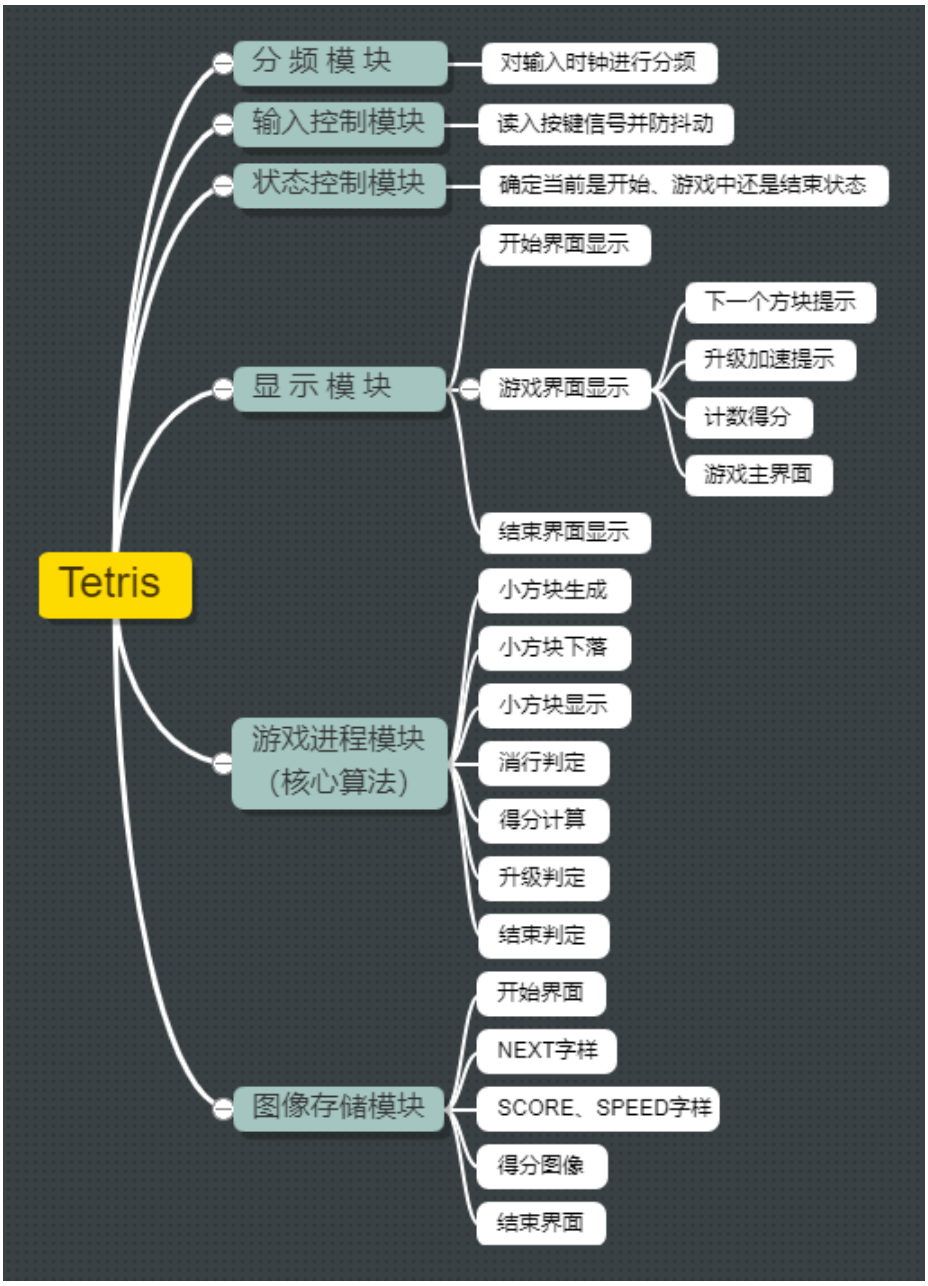

Figure 2　Tetris 模块划分

Table 2　模块详细

模块名	源文件名	功能
顶层实体	Tetis	实例化并连接其他模块
分频器	my_pll	将 50MHz 的系统时钟处理为 25MHz 的 VGA 时钟信号
防抖动（输入控	deboucer	读入按键（游戏重置键 rst_n、方块左移键

制）	v_speed_debouncer			left、方块右旋键 rotaeR、方块右移键 right、verticalspeed）信号，并对信号进行防抖动处理。本游戏另设置向下加速按键，但由于 DE2i-150 开发板共计 4 个按钮，如有必要，可将该按键移植到按钮数量更多的开发板。
系统状态机及其显示控制	automaton			确定当前是开始、游戏中还是结束状态
	vga_select_module			根据 automaton 提供三大信号选择当前是哪种状态的输出
开始	start_vga_control			控制开始页界面的输出
	start_sync_module			显示开始页
	start_instance_name			开始页的图像信息
游戏中	loading_happen	非字模	display_border	游戏进行中，俄罗斯方块主算法
			display_little_square	
			display_next_square	
			display_moving_square&square_gen	
		字模	display_score&n 系列	
			show_next&next_rom	
			show_score&score&spscore	
	game_display			颜色设置
	game_sync_module			显示游戏界面
结束	over_vga_control			控制结束页界面的输出
	over_sync_module			显示结束页
	over_view_mem			结束页的图像信息

3. VGA 时序分析

3.1 VGA 扫描原理

VGA 显示器扫描方式分为逐行扫描和隔行扫描:

逐行扫描是从屏幕左上角第一个点开始，从左向右逐点扫描，每扫描完一行，电子束回到屏幕的左边下一行的起始位置，在这期间，CRT 对电子束进行消隐，每行结束时，用行同步信号进行同步；当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐，开始下一帧。

隔行扫描是指电子束扫描时每隔一行扫一线，扫完一屏后再返回来扫描剩下的线，隔行扫描的显示器闪烁快速，可能会使使用者眼睛疲劳。

本项目中采用的是逐行扫描的方式。

3.2 行、帧同步

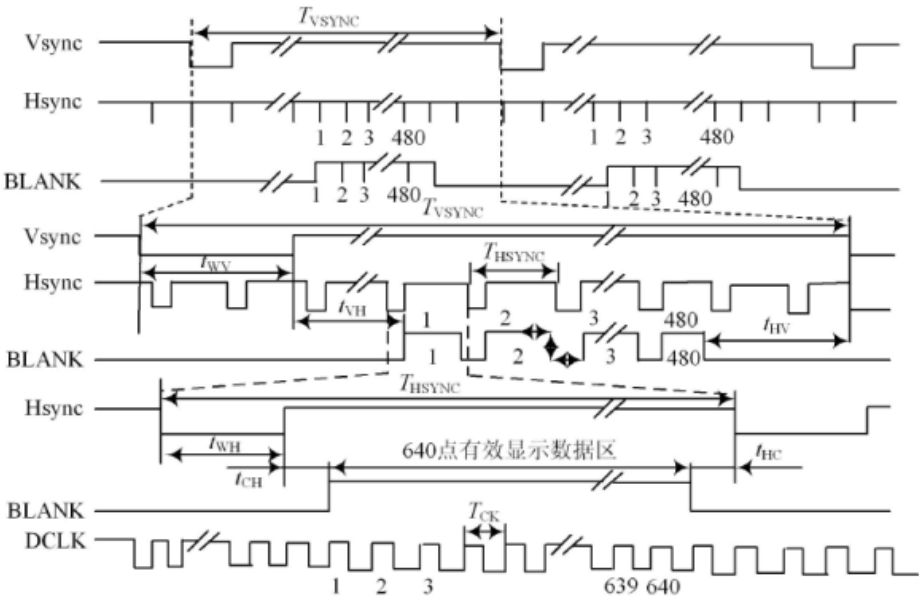

Figure 3 VGA 640*480 60Hz **时序图**

VGA 中定义行时序和帧时序都需要同步脉冲（a 段），显示后沿（b 段）、显示时序段（c 段）和显示前沿（d 段）四部分。VGA 工业标准显示模式要求：行同步、帧同步都为负极性，即同步脉冲要求是负脉冲。

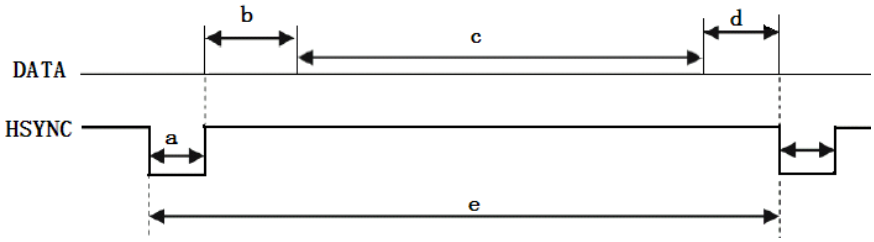

Figure 4 **行同步时序**

每一行都有一个负极性行同步脉冲（a 段），是数据行的结束标志，同时也是下一行的开始标志。在同步脉冲之后为显示后沿（b 段），在显示时序段（c 段）显示器为亮的过程，RGB 数据驱动一行上的每一个像素点，从而显示一行。在一行的最后为显示前沿（d 段）。在显示时间段之外没有图像投射到屏幕是插入消隐信号。同步脉冲、显示后沿和显示前沿都是在行消隐间隔内，当消隐有效时，RGB 信号无效，屏幕不显示数据。

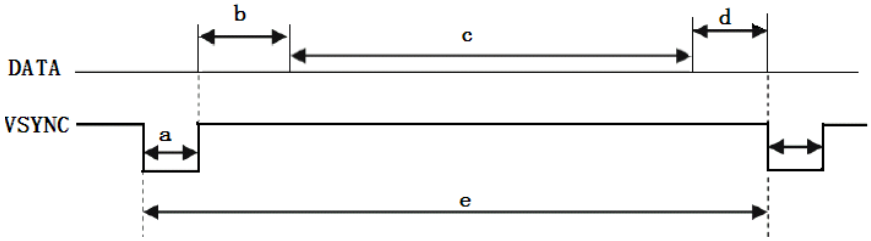

Figure 5 **帧同步时序**

每一帧都有一个负极性行同步脉冲（a 段），是数据帧的结束标志，同时也是下一帧的开始标志。在同步脉冲之后为显示后沿（b 段），在显示时序段（c 段）显示器为亮的过程，

RGB 数据驱动一帧上的每一个像素行，从而显示一帧。在一帧的最后为显示前沿（d 段）。在显示时间段之外没有图像投射到屏幕是插入消隐信号。同步脉冲、显示后沿和显示前沿都是在帧消隐间隔内，当消隐有效时，RGB 信号无效，屏幕不显示数据。

3.3 VGA 显示区域

显示模式	时钟（MHz）	行时序（列数）					帧时序（行数）				
		a	b	c	d	e	a	b	c	d	e
640*480*60	25.175	96	48	640	16	800	2	33	480	10	525
640*480*75	31.5	64	120	640	16	840	3	16	480	1	500
800*600*60	40	128	88	800	40	1056	4	23	600	1	628
800*600*75	49.5	80	160	800	16	1056	3	21	600	1	625
1024*768*60	65	136	160	1024	24	1344	6	29	768	3	806
1024*768*75	78.8	176	176	1024	16	1392	3	28	768	1	800
1280*1024*60	108	112	248	1280	48	1688	3	38	1024	1	1066
1280*800*60	83.64	136	200	1280	64	1680	3	24	800	1	828
1440*900*60	106.47	152	232	1440	80	1904	3	28	900	1	932

Figure 6　VGA 显示标准

项目采用 640×480×60 的显示模式，由此可得显示区域示意图:

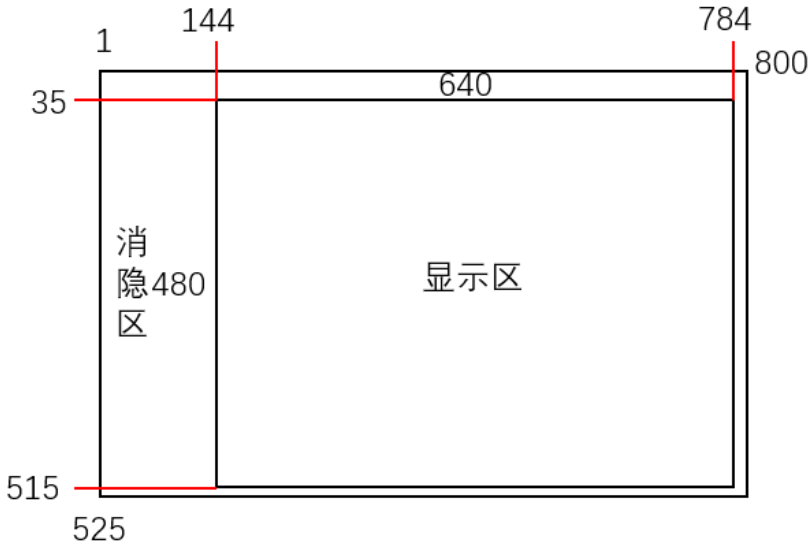

Figure 7　屏幕显示有效区域示意图

其中中间 640×480 的部分为屏幕的有效显示区域，在实现的过程中需要对这里进行设置。

3.4 时钟频率

项目采用 640×480×60Hz 的显示模式，其中 640 为列数，480 为行数，60 为帧频。由表可知，每帧有 525 行，则行频为 525×60Hz = 31500Hz。每行有 800 个像素点，则单点需要的时钟频率为 800×31500Hz = 25.2MHz ≈ 25MHz。

在实现的过程中需要设置 pll 锁相环对时钟进行分频，整合出 25MHz 的时钟信号。

4. 游戏流程

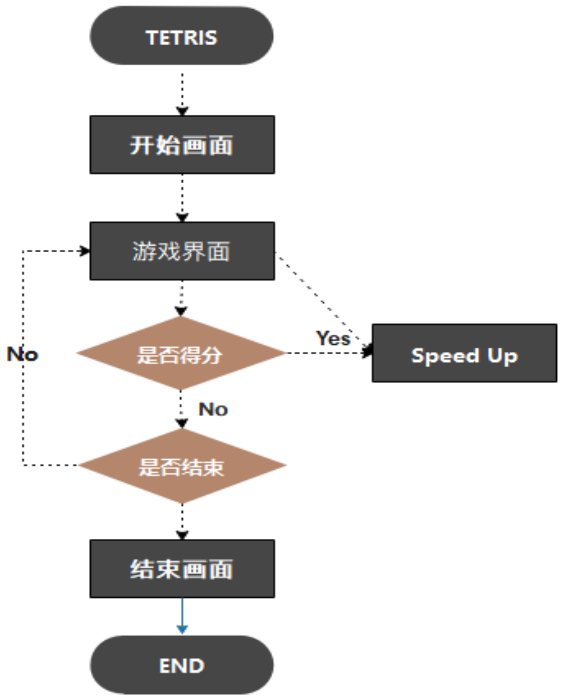

Figure 8 游戏流程图

四、 实验方法和实验步骤

1. 顶层实体

1.1 Tetris

Table 3 顶层的输入输出

I/O 名	意义
input sys_clk	系统输入时钟（DE2i-150）为 50MHz
input rst_n	系统重置信号输入
input right	方块组右移键信号
input left	方块组左移键信号
input rotateR	方块组顺时针旋转信号
output[7:0] VGA_G	VGA 中 R G B 信号的输出，决定了屏幕上颜色的显示
output[7:0] VGA_R	
output[7:0] VGA_B	
output VGA_CLK	VGA 时钟输出（25MHz）
output VGA_VSYNC	行同步信号输入
output VGA_HSYNC	场同步信号输入
output VGA_SYNC_N	低电平有效
output VGA_BLANK_N	无效时间
input verticalspeed	方块四倍速下落信号（缺少按钮）

此处定义个整个项目的输入和输出引脚，并且将其他模块创建实例后连接起来形成整个系统。

2. 游戏状态模块

2.1 automaton.& vga_select_module

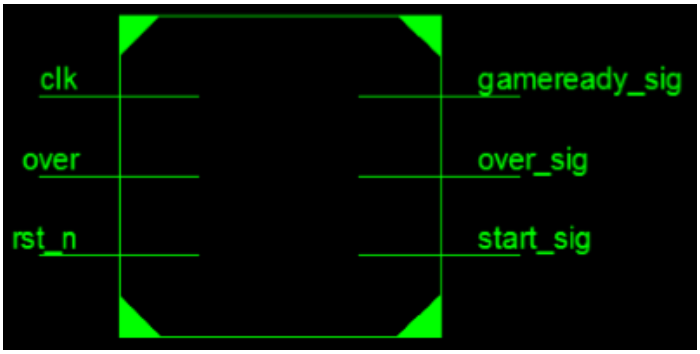

Figure 9 automaton

Table 4 automaton 模块控制状态机

I/O 名	意义
input clk	VGA 时钟
input rst_n	系统重置信号输入
input over	游戏结束信号输入（由 loading_happen 提供 game over 信号）
output gameready_sig	游戏准备信号输出（作为 start_vga_control_model 的输入信号）
output start_sig	游戏开始信号输出(作为 loading_happen 的输入信号)
output over_sig	游戏结束信号输出(作为 over_vga_contol 的输入信号)

- automaton 状态机为系统三大状态的状态判断和处理模块:
  - 根据 rst_n（或系统第一次启动）信号发出 gameready_sig 信号，初始模块响应该信号输出开始画面;
  - 计时五秒后，发出 start_sig 信号，游戏开始并进行，同步输出相应的游戏界面;
  - 接收核心算法模块“loading_happen”提供的 over 信号，判断游戏终止，结束模块响应该信号并输出结束画面。
- vga_select_module 根据 automaton 提供三大信号选择当前是哪种状态的输出。

2.2 start_vga_control_module & start_sync_module.& start_instance_name

- 由 automaton 模块的输出信号 gameready_sig 触发 start_vga_control_module 对开始界面的输出;
  - start_instance_name 中存储着开始界面的图像信息;
  - start_sync_module 负责开始动画的显示。

2.3 loading_happen & game_sync_module（此模块为核心算法入口，将在第五小节详细介绍）

- 由 automaton 模块的输出信号 start_sig 触发 loading_happen 游戏的开始;
  - game_display 负责游戏界面的显示;
  - game_sync_module 负责游戏进行中界面的输出。

2.4 over_vga_control_module & over_sync_module & over_view_mem

- 由 automaton 模块的输出信号 over_sig 触发 over_vga_control 对结束画面的输出；
  - over_view_mem 中存储着结束画面的图像信息；
  - over_sync_module 负责结束动画的显示。

3. 分频

3.1 my_pll

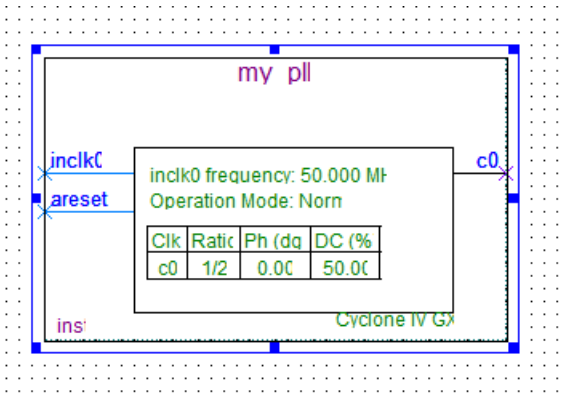

Figure 10 **锁相环** bdf

根据 VGA 时序分析，我们使用 IP 核建立 pll 模块，对 50MHz 的系统时钟进行处理。默认选中 areset 项，选择 clk c0 输出 25MHz 的 VGA 时钟频率。设计 IP 核时选中 my_pll_inst.v（用于时序仿真）、my_pll_bb.v 和 my_pll_bsf.v 文件，其 bdf 文件如上。

4. 防抖动

4.1 deboucer

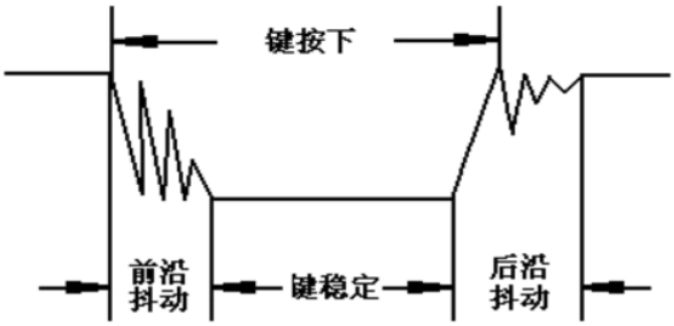

Figure 11 **抖动的产生**

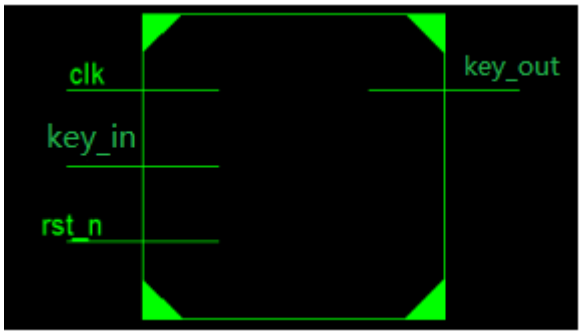

Figure 12 deboucer

Table 5 deboucer

input clk	输入时钟
input rst_n	系统 reset 信号输入
input key_in	按键信号输入
output key_out	按键信号输出

说明：消除按键按下以及抬起时所带来的抖动。

4.2 v_speed_deboucer（这个功能由于我们的按键比较少，所以未能演示）

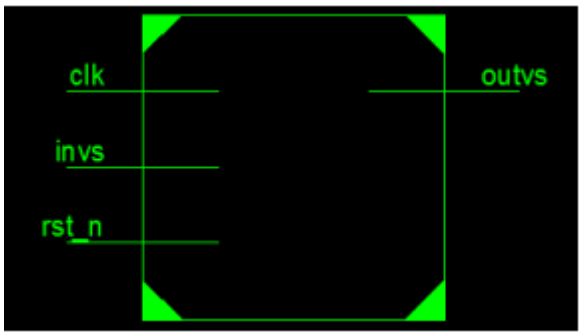

Figure 13 v_speed_deboucer

Table 6 v_speed_deboucer

input clk	输入时钟
input rst_n	系统 reset 信号输入
input invs	输入垂直方向下落加速信号
output outvs	输出垂直方向下落加速信号

5. 游戏算法

5.1 Loading_happen

Table 7 loading_happen

I/O	**意义**
Input start_sig	游戏开始信号
Input clk	时钟信号
Input rst_n	重启信号

Input move_right	下落方块右移
Input move_left	下落方块左移
Input [15:0] enable_moving	下落方块形状
Input verticalspeed_out	下落加速
Output [10:0] h	下落方块水平像素位置
Output [10:0] v	下落方块垂直像素位置
Output [499:0] enable_little	游戏界面存储
Output loading_square	生成下一个方块
Output [8:0] little_square_num	下落中方块坐标
Output over_out	游戏结束
Output [15:0] score_out	计分结果
Output level_up	游戏升级信号

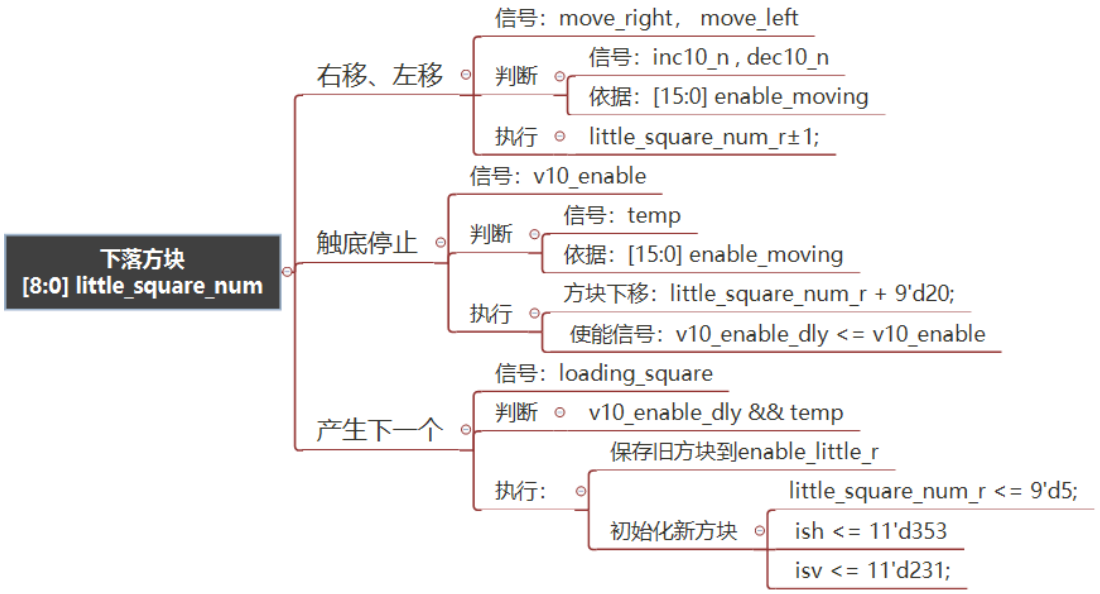

Figure 14 下落方块

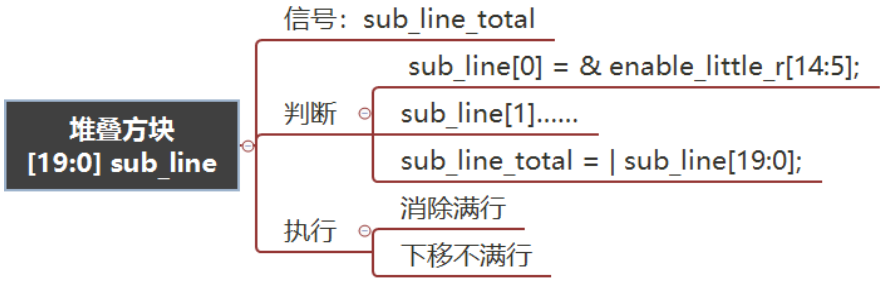

Figure 15 堆叠方块

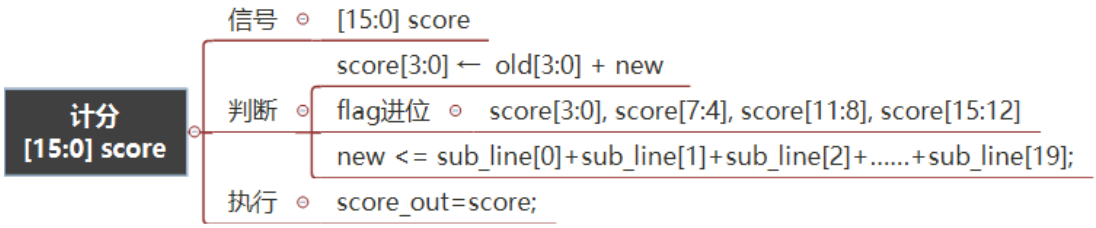

Figure 16 计分

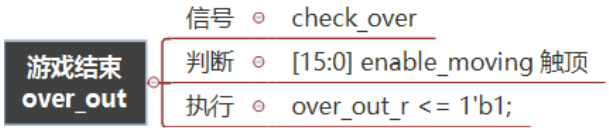

Figure 17 游戏结束判断

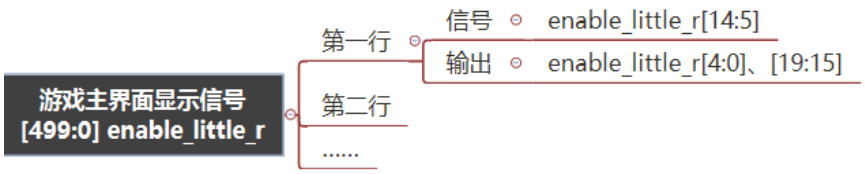

Figure 18 显示信号

5.2 game_display

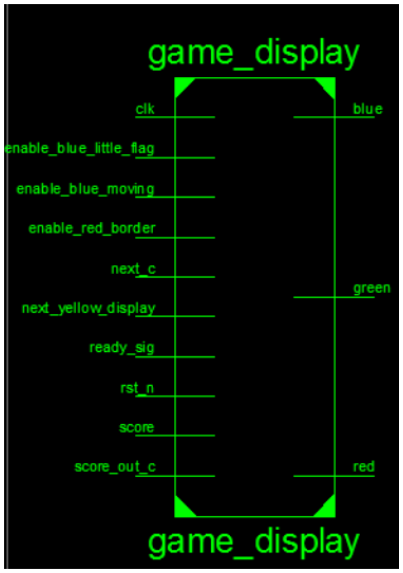

Table 8　game_display

I/O	意义
input clk	时钟信号
input rst_n	系统 reset 信号
input ready_sig	游戏准备状态
input enable_red_border	边框信号
input enable_blue_moving	下落小方块显示信号
input enable_blue_little_flag	静态小方块显示信号
input next_yellow_display	下一小方块显示信号
input next_c	下个方块组
input score_out_c	score 图片
input score	得分
output red	处理完的 rgb 信号
output green	
output blue	

这个模块根据内部模块状态，设置颜色，输出 RGB 信号，如：

- 分数显示：Blue + Green = Cyan
- 下一个方块：Green + Red = Yellow
- 静止的方块：Blue + Green = Cyan

```
isgreen <= score|enable_red_border | next_yellow_display
        |enable_blue_little_flag| score|next_c;
isred <= next_yellow_display|next_c|score_out_c;//用于显示next字符
isblue <= score|enable_blue_little_flag|enable_blue_moving|next_c;
```

5.3 game_sync_module——游戏界面输出

Table 9　game_sync_module

I/O	意义

Input clk	VGA 时钟
Input rst_n	重启信号
Output [10:0] col_addr_sig	列地址信号
Output [10:0] row_addr_sig	行地址信号
Output hsync	水平同步信号
Output vsync	水平消隐信号
Output ready_sig	RGB 输出使能信号

6. 游戏进行中的显示

6.1 Display_boder

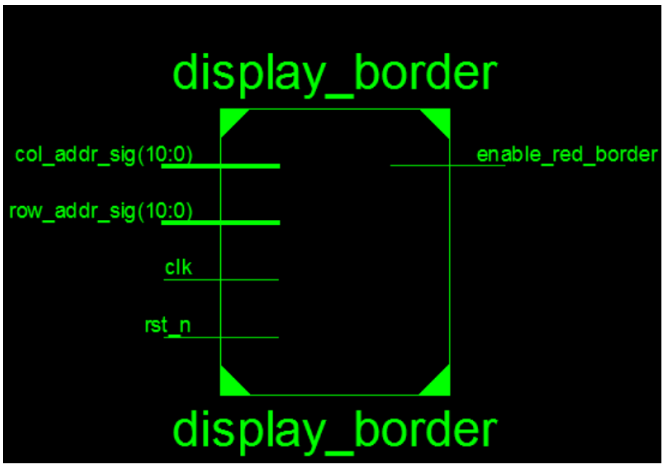

Figure 19 display_boder

Table 10 display_border

I/O	意义
input clk	VGA 时钟
input rst_n	系统 reset 信号
input[10:0] col_addr_sig	列地址信号
input[10:0] row_addr_sig	行地址信号
output enable_red_border	边框信号

这个模块根据输入的行、列地址，判断其所对应的像素点是否在游戏边框的位置。若是则输出边框信号为 1，否则输出边框信号为 0。

6.2 display_little_square

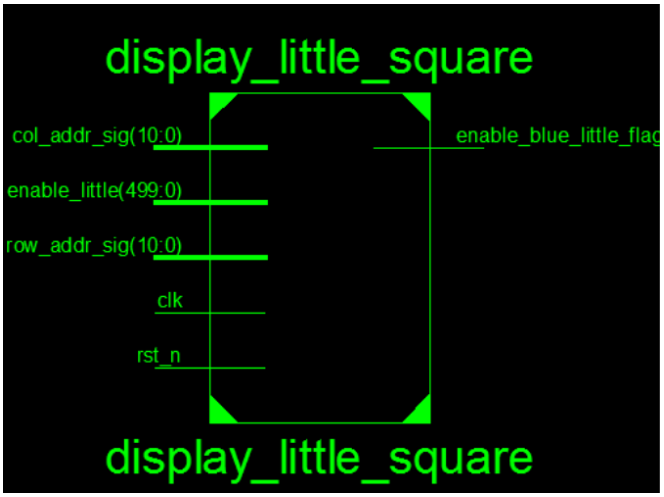

Figure 20 display_little_square

Table 11 display_little_square

I/O	意义
input clk	VGA 时钟信号
input rst_n	系统 reset 信号
input[10:0] col_addr_sig	列地址信号
input[10:0] row_addr_sig	行地址信号
input[499:0] enable_little	游戏区域小方块状态信号
output enable_blue_little_flag	小方块显示信号

这个模块根据输入的行、列地址，以及游戏区域小方块状态，为每个小方块绘制 20*20 的像素区域。

6.3 display_moving_square

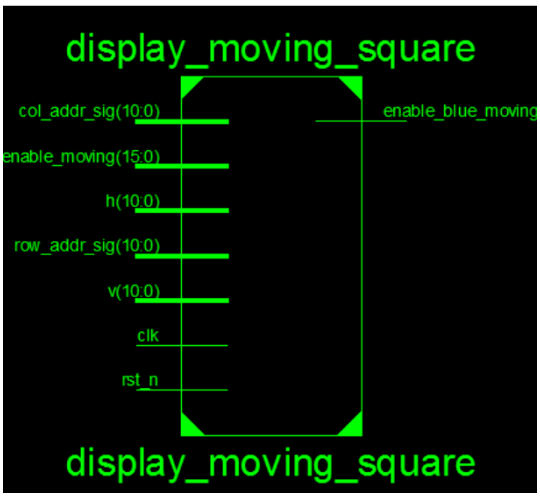

Figure 21 display_moving_square

Table 12 display_moving_square

I/O	意义
input clk	时钟信号
input rst_n	系统 reset 信号
input[10:0] col_addr_sig	列地址信号

input[10:0] row_addr_sig	行地址信号
input[10:0] h	行像素坐标
input[10:0] v	列像素坐标
input[15:0] enable_moving	下落小方块信号
output enable_blue_moving	下落小方块显示信号

这个模块根据输入的行、列地址，以及游戏区域内下落小方块之间的相对位置状态和 h、v 表示的下落方块整体的绝对位置状态，判断行、列地址所对应的像素点是否落在下落的小方块内。若是则输出下落小方块显示信号为 1，否则输出下落小方块显示信号为 0。

6.4 square_Gen

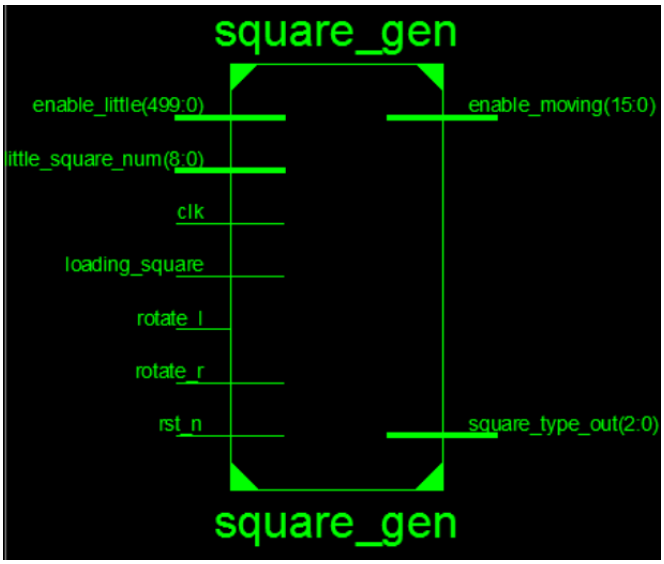

Figure 22 square_gen

Table 13 square_gen

I/O	**意义**
input clk	VGA 时钟信号
input rst_n	系统 reset 信号
input rotate_r	右旋信号
input rotate_l	左旋信号（未使用）
input loading_square	加载方块信号
input[499:0] enable_little	游戏区域小方块状态信号
input[8:0] little_square_num	下落方块组坐标
output[15:0] enable_moving	下落方块组形状坐标
output[2:0] square_type_out	下一方块组形状编号

这个模块的功能有二：一是如果输入了右旋信号，那么根据游戏区域小方块状态信号、下落方块组坐标以及旋转后方块组坐标检测冲突，输出旋转后或没有旋转的下落方块组形状坐标；二是如果输入了加载方块信号，则根据生成的随机数输出下一方块组形状编号。每个初始方块组形状编号对应一个方块组形状坐标。

6.5 display_next_sauqre

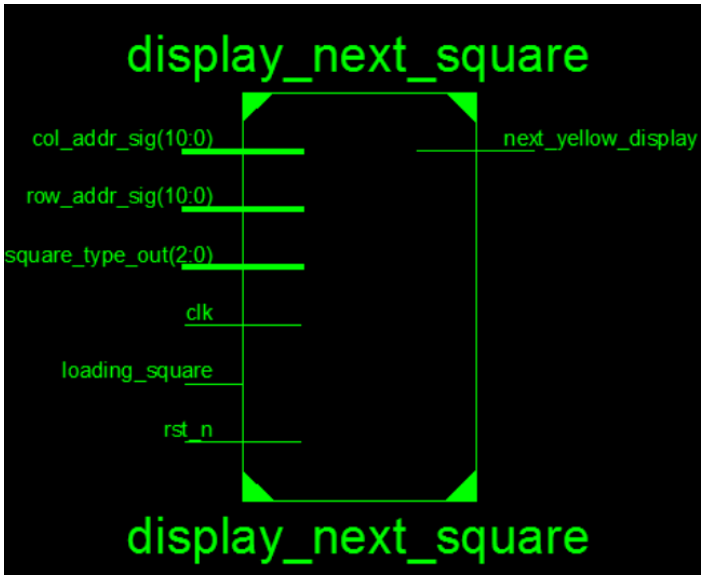

Figure 23display_next_square

Table 14 display_next_square

input clk	**输入 时钟信号**
input rst_n	输入 系统 reset 信号
input[10:0] col_addr_sig	输入 列地址信号
input[10:0] row_addr_sig	输入 行地址信号
input loading_square	输入 加载方块信号
input[2:0] square_type_out	输入 方块组形状编号
output next_yellow_display	输出 下一小方块显示信号

这个模块根据输入的行、列地址，以及下以方块组形状编号，判断地址所对应的像素点是否落在下一小方块显示区域内。若是则输出下一小方块显示信号为 1，否则输出下一小方块显示信号为 0。

7. 游戏中的字模显示

首先，制作开始界面、结束界面、“next”字样、“score”字样、“speed up”字样和 0~9 的阿拉伯数字的黑白 bmp 文件。其中“score”和“speed up”在同一位置显示，故两张图片的大小一致。0~9 的阿拉伯数图片大小也均一致。

然后，通过 mif 生成器，将 bmp 文件以解析为黑 0 白 1 的数据初始化文件.mif 文件。

最后，根据每个 mif 文件的大小为其新建 IP 核管理器，后续我们可以通过相应的封装代码对其进行实例化操作。

7.1 display_score & n_0~n_9

Table 15 display_score

I/O 名	意义
input clk	输入 VGA 时钟
input rst_n	输入 重置信号
input[10:0] col_addr_sig	输入 列地址信号
input[10:0] row_addr_sig	输入 行地址信号
input[15:0] score_out	输入 计分结果

output color_score_out	输出 加分显示

本实验最高得分为 9999，故将计分区划分为四，分别代表个、十、百、千位。

- display_score 负责计数得分，并更新显示信息；
  - n_0 到 n_9 分别存储着阿拉伯数字的图像信息，用于显示得分。

7.2 show_next & next_rom（和 display_score 相似）

- show_next 负责在屏幕左上方显示“next”字样
  - next_rom 存储着“next”图像信息。

7.3 show_score & score & spscore（和 display_score 相似，另有 levelup_sig 为升级信号）

- show_score 负责显示无升级状态时的“SCORE”和升级后的“SPEED UP！”字样
  - “score”和“spscore”中分别存储着相关单词的图像信息。

8. 管脚约束

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
VGA_B[7]	Output	PIN_C22	7	B7_N1	PIN_C22	2.5 V (default)		16mA (default)	2 (default)
VGA_B[6]	Output	PIN_G25	7	B7_N0	PIN_G25	2.5 V (default)		16mA (default)	2 (default)
VGA_B[5]	Output	PIN_A23	7	B7_N1	PIN_A23	2.5 V (default)		16mA (default)	2 (default)
VGA_B[4]	Output	PIN_F24	7	B7_N0	PIN_F24	2.5 V (default)		16mA (default)	2 (default)
VGA_B[3]	Output	PIN_C23	7	B7_N1	PIN_C23	2.5 V (default)		16mA (default)	2 (default)
VGA_B[2]	Output	PIN_B25	7	B7_N1	PIN_B25	2.5 V (default)		16mA (default)	2 (default)
VGA_B[1]	Output	PIN_C24	7	B7_N1	PIN_C24	2.5 V (default)		16mA (default)	2 (default)
VGA_B[0]	Output	PIN_E24	7	B7_N0	PIN_E24	2.5 V (default)		16mA (default)	2 (default)
VGA_BLANK_N	Output	PIN_F25	7	B7_N0	PIN_F25	2.5 V (default)		16mA (default)	2 (default)
VGA_CLK	Output	PIN_D27	7	B7_N0	PIN_D27	2.5 V (default)		16mA (default)	2 (default)
VGA_G[7]	Output	PIN_B22	7	B7_N1	PIN_B22	2.5 V (default)		16mA (default)	2 (default)
VGA_G[6]	Output	PIN_A22	7	B7_N1	PIN_A22	2.5 V (default)		16mA (default)	2 (default)
VGA_G[5]	Output	PIN_F21	7	B7_N1	PIN_F21	2.5 V (default)		16mA (default)	2 (default)
VGA_G[4]	Output	PIN_A21	7	B7_N1	PIN_A21	2.5 V (default)		16mA (default)	2 (default)
VGA_G[3]	Output	PIN_K19	7	B7_N2	PIN_K19	2.5 V (default)		16mA (default)	2 (default)
VGA_G[2]	Output	PIN_A20	7	B7_N2	PIN_A20	2.5 V (default)		16mA (default)	2 (default)
VGA_G[1]	Output	PIN_C20	7	B7_N1	PIN_C20	2.5 V (default)		16mA (default)	2 (default)
VGA_G[0]	Output	PIN_D20	7	B7_N1	PIN_D20	2.5 V (default)		16mA (default)	2 (default)
VGA_HSYNC	Output	PIN_B24	7	B7_N1	PIN_B24	2.5 V (default)		16mA (default)	2 (default)
VGA_R[7]	Output	PIN_C19	7	B7_N2	PIN_C19	2.5 V (default)		16mA (default)	2 (default)
VGA_R[6]	Output	PIN_B19	7	B7_N2	PIN_B19	2.5 V (default)		16mA (default)	2 (default)
VGA_R[5]	Output	PIN_E19	7	B7_N1	PIN_E19	2.5 V (default)		16mA (default)	2 (default)
VGA_R[4]	Output	PIN_E18	7	B7_N1	PIN_E18	2.5 V (default)		16mA (default)	2 (default)
VGA_R[3]	Output	PIN_A18	7	B7_N2	PIN_A18	2.5 V (default)		16mA (default)	2 (default)
VGA_R[2]	Output	PIN_B18	7	B7_N2	PIN_B18	2.5 V (default)		16mA (default)	2 (default)
VGA_R[1]	Output	PIN_C18	7	B7_N2	PIN_C18	2.5 V (default)		16mA (default)	2 (default)
VGA_R[0]	Output	PIN_A17	7	B7_N2	PIN_A17	2.5 V (default)		16mA (default)	2 (default)
VGA_SYNC_N	Output	PIN_AH20	4	B4_N1	PIN_AH20	2.5 V (default)		16mA (default)	2 (default)
VGA_VSYNC	Output	PIN_A24	7	B7_N1	PIN_A24	2.5 V (default)		16mA (default)	2 (default)
left	Input	PIN_AF30	5	B5_N2	PIN_AF30	2.5 V (default)		16mA (default)	
right	Input	PIN_AA26	5	B5_N2	PIN_AA26	2.5 V (default)		16mA (default)	
rotateR	Input	PIN_AE25	5	B5_N2	PIN_AE25	2.5 V (default)		16mA (default)	
rst_n	Input	PIN_AE26	5	B5_N2	PIN_AE26	2.5 V (default)		16mA (default)	
sys_clk	Input	PIN_AJ16	4	B4_N2	PIN_AJ16	2.5 V (default)		16mA (default)	
<<new node>>									

Figure 24 管脚约束

五、 测试仿真

1. 时序仿真

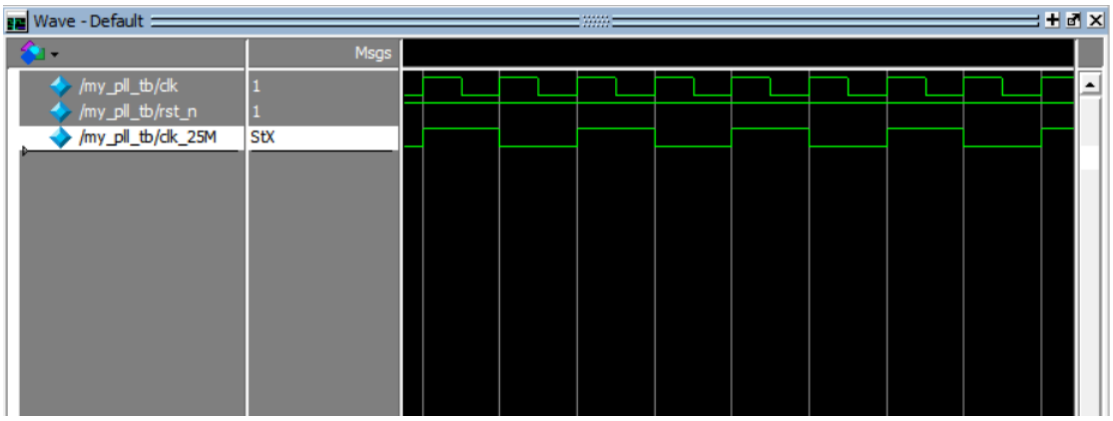

Figure 25 **时序仿真**

仿真 IP 核时，需要加入 altera_mf.v 和 220model.v。

如图，此段截图中输出频率 clk_25M（即分频后的 VGA 时钟 c0）为输入频率 clk（即系统输入时钟 sys_clk）的一半。成功将 50MHz 的系统时钟分频为 25MHz。

2. automaton

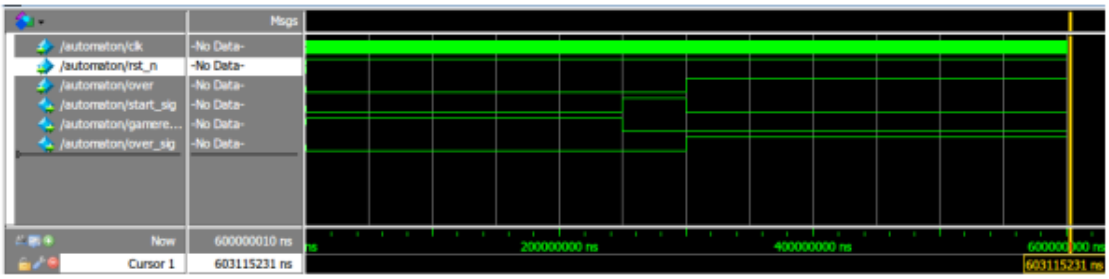

Figure 26 **总控状态机仿真**

如图，当模拟出的 loading_happen 给出 over 信号，automaton 发出 over_sig 信号，该信号触发结束页面的显示。gameready 信号触发开始页面的显示，start_sig 和 gameready 同时产生，但是它持续 5s，5s 之后开始游戏。

3. deboucer

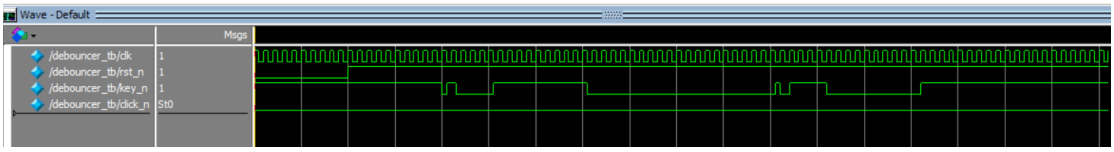

Figure 27 **普通按键消抖**

如图，deboucer 能够消除抖动时间（5ms）内的模拟出的按键抖动。

4. v_speed_deboucer

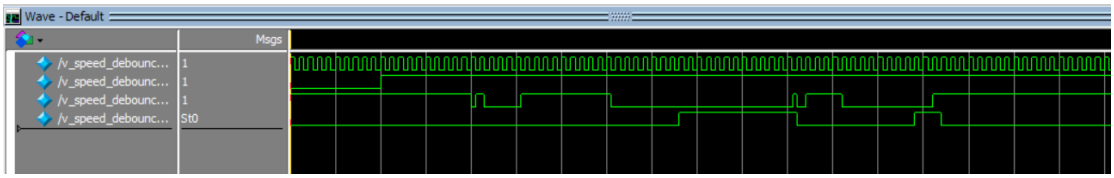

Figure 28 **下落按键消抖**

六、 实验结果和演示

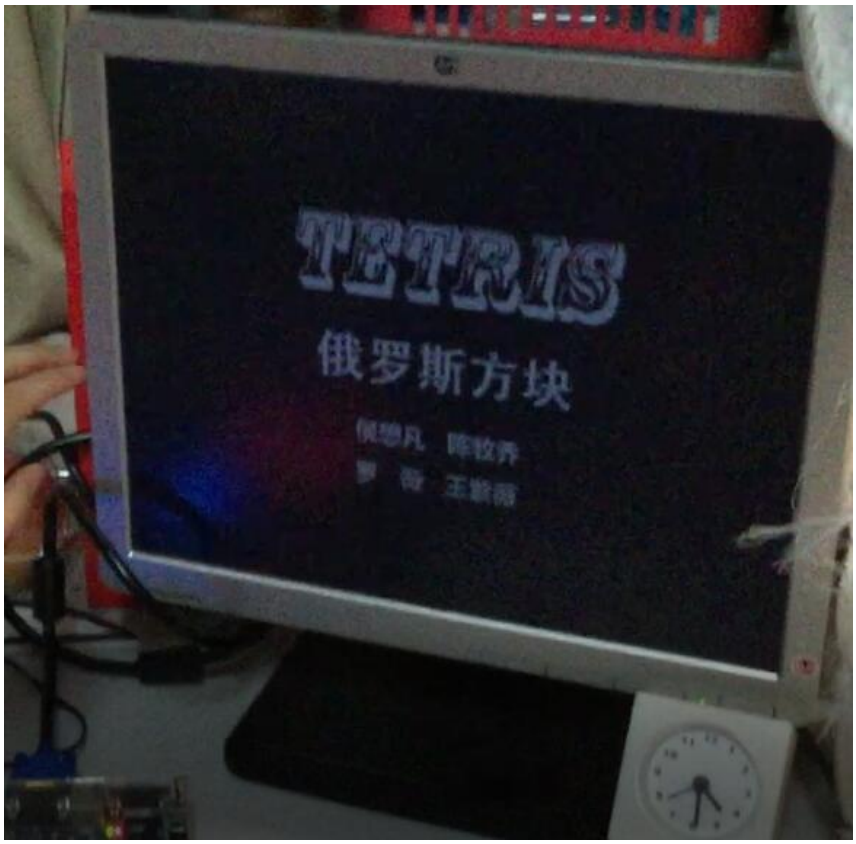

Figure 29 开始界面

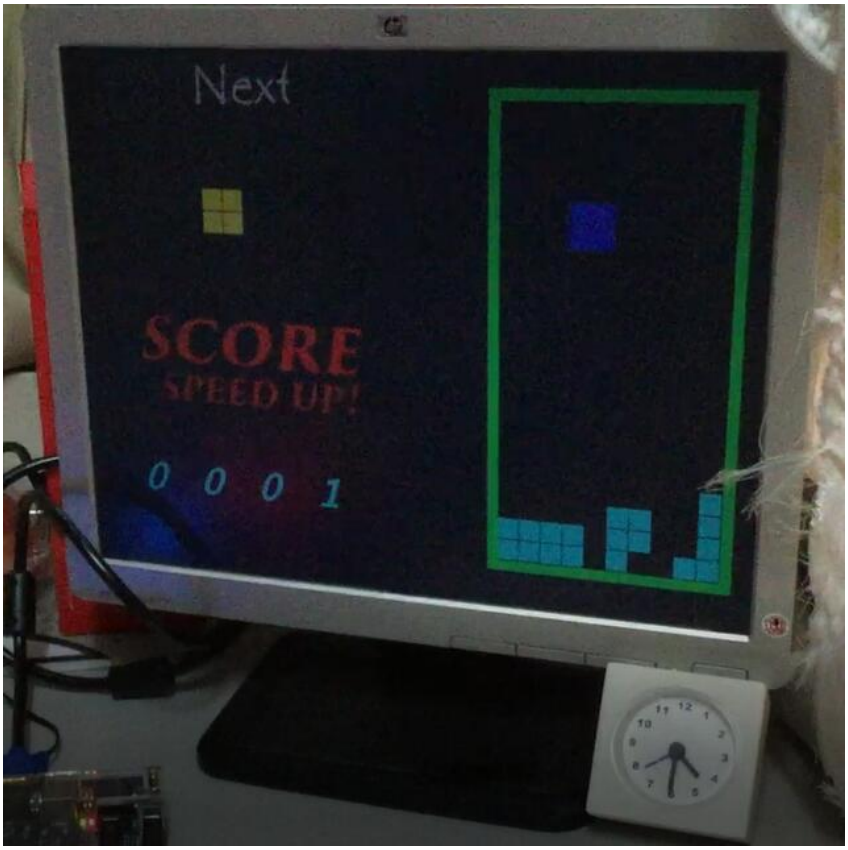

Figure 30 游戏中

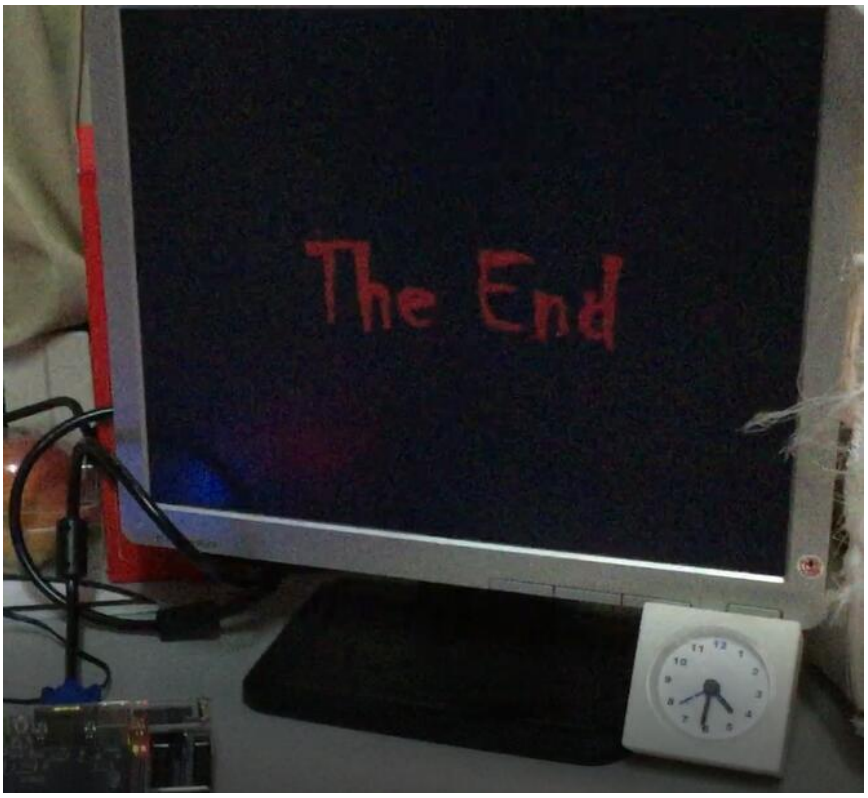

Figure 31 结束界面

七、 实验中遇到的错误

1. 调整图片大小至引脚所能到达的范围

起初设计的 640*480 的图片，但是 ROM 能提供范围不够存储

2. CIRTICAL WARNING: memory 必须一样大

根据 mif 文件设计 IP 核，words 数量只粗略地接近，报出 critical warning。将数值设置为与 mif 文件完全一致即可。

3. 引脚设计——push botton

根据俄罗斯方块游戏的要求，一般需要左移、右移、向下加速、左旋、右旋五个游戏控制按钮。另加上必须的重置按钮 rst_n，初期设计了开始游戏按钮。共计 7 个按钮。

但 DE2i-150 只提供了 4 个 push button。

为适应硬件条件，最终我们取左移、右移、右旋和加速的游戏功能，rst_n 使用一个 switch 开关键。

4. mif 手动填充数据量太大

找到自动解析图片并生成 mif 文件的软件，免去了手动填写 mif 文件的工作。

5. CIRTICAL WARNING:Create a default TimeQuest SDC file

根据用户手册指导，时间设置是至关重要的。我们为他创建一个同名于项目的 sdc 文件，并且编辑相应的代码即可。

八、 心得、体会

1. CMQ

这个项目在开始之前几乎是不可能完成的任务，虽然之前的课程中我们已经做过一些实验，但是实验的步骤以及核心代码大多数都是老师提供的，我们所做的是熟悉板子的开发流程并将程序成功跑出来，这当中可发挥创新的地方并不多。而最后的项目则没有现成的操作步骤，从选题到设计实现都需要我们自己独立完成。这就需要对之前课程的内容非常熟悉并能够灵活运用，是非常考验课程学习效果的。

这是一个很综合的项目，包括游戏逻辑、VGA 显示、分频、硬件配置等等好几个模块，需要合理的小组成员分工与合作，想做好并不是件容易的事。在初期总是有些急于求成，想尽快地跑出个样子来，在一些代码的编写上有些不过脑子，意气用事，结果几乎是颗粒无收。因此一定要静下心来，仔细地弄清楚每个模块的功能，也要特别注意模块之间接口的衔接，做到心中有数，才能在此基础上进行进一步的发挥创造。对于一些自动生成的部分，比如字模、ROM/RAM、PLL，很难做到把生成的代码每一行都弄懂，那就要把重点放在生成过程中每一个填入的参数上，在早期没能完全理解，后来才发现其重要性。特别是 PLL，它是整个项目运行的基础，它的配置直接影响到显示器是否能有显示。而 VGA 显示是一项十分琐碎繁杂的工作，代码与字模要做到十分精准的匹

配，否则就会失之毫厘谬以千里——调试过程中生成的各种奇形怪状的图案就是最直观的教训。

在一次次碰壁的过程中我对 VGA 显示原理以及硬件编程机制有了更深入的理解。我们的 VGA 项目的最终目标就是确定 25MHz 时钟下每个像素点要显示的内容——一个看上去可能不动的像素点，实际上一秒钟刷新了 60 次。而硬件编程，与 C 语言编程的不同，是在于它是基于时钟的。25MHz 同步时钟是它的命脉，在它的节奏下每个模块的每个小状态紧密配合，共同组成一整个屏幕的大状态。同步、并行，颇像一部交响乐，不同乐器的不同的音色、音调、音高密切配合，共同奏出一曲华伟的篇章。

2. WZW

整个项目从选题，设计到最后通宵调试，录制展示视频，一共花了将近一个月的时间，每一步的尝试和摸索都被一一记下，回看时也清楚的看到我们走了很多弯路，都最终构成这门选修课最后阶段的完整回忆。

我和另一个组员侯思凡同学最初决定做俄罗斯方块这个游戏，既考虑到游戏规则较为熟悉，同时可以合理利用 DE2-i150 的按键。但在资料搜集阶段我们花了很大的功夫，也只找到了关于 DE2 板子的些许不完整的可借鉴资料。我们希望将整个项目的游戏算法和显示分成两个打的模块分工完成，但最初从无到有的阶段，输入输出信号的确定都是未知且茫然的。游戏部分的算法我们去读用其他语言实现俄罗斯方块项目的代码，如何保存方块的坐标，如何实时显示每个小方块。我们在 GitHub 等渠道学习大神的博客日志，甚至给在美国的做过此类项目的陌生人发邮件问问题。好不容易写出编译不报错的一部分程序，等每周四周五尝试下板遇到“无信号”、黑屏等现象，无从得知问题出在哪里，也只能你一言我一语抓住一切可能改变、调试。直到最后一周之前才终于在显示器上看见方块，心里一块大石头才终于落地。

虽然可以显示，但这个时候我们的字模显示基本都有问题，加速功能也还没有实现。但好在这个时候整体框架确定，便再次分工解决图片的显示和加速功能。记得最后一天验收之前还有一门考试，考完试每个人都不敢松懈吃完饭便开始解决最后开机画面、加速按键的问题，最终无差错运行时抬头一看外面天都亮了，已经是接近早上五点。

过程充满坎坷，但结局很圆满。从搞不懂代码的执行过程到后来修改、优化，从满屏幕乱七八糟的显示到最后能随心所欲的修改图片形式和大小，我们走的缓慢艰辛但却脚踏实地。半个多学期的时间根本不长，做成这样一个项目也绝非易事，因此我格外珍视这次经历和我我的组员兼室友为此付出的所有努力。

3. LW

本次项目从纯硬件的角度实现了俄罗斯方块小游戏，实验采用 FPGA 设计，运用 Verilog 语言描述数字电路，用 VGA 显示的方法，在 DE2i-150 开发板上进行。以下是我在本次实验过程中的收获。

1) 代码模块化

在用 Verilog 语言进行建模的时候，适当地对要完成的模块进行划分是一个很好的建模习惯，在保证功能要求的满足的前提下，能够使自己的代码容易理解、维护，并减少一些容易忽视的错误。

2）硬件设计体会

以往的计算机专业实验大部分是进行软件的编制和调试，与实际应用中的硬件电路相脱节。硬件的测试比软件麻烦，在代码大致编写完成到了调试阶段时，需要经历全编译和下板的过程，测试一次需要花费几分钟。因此在修改程序时细致一些，否则出错将浪费很多时间。在下板之前先进行仿真检查程序，就可以有效降低出错率。

3）实时记录阶段成果

本次项目从初期设计到编写代码到下板测试，耗时较长。因此每当我们完成一个阶段性的计划就应该记录下来，或者在一天的工作结束后把遇到的问题和今后的方向写进日志，以供日后查阅。尤其到了代码测试后期，当遇到错误无法解决时，可以从之前的工作记录中找灵感。

4）团队分工合作

当项目规模较大时，例如这次的大作业，团队中的任务分工、交流与合作就显得尤为重要。我们付出时间和精力完成各自的部分，在项目全程一起学习讨论，互相帮助队友，直到最终完成。在进行团队合作的时候，还要耐心听取每个成员的意见，使我们的组合更加默契、高效。

5）善于利用资源

开发板设计、verilog 语言等都是在硬件设计课程之前从未接触到的内容，而课堂学习时长有限，老师讲授和我们吸收的知识点也不是很多，再加上硬件实验需要下板的要求，我们实验进行得并不是一帆风顺。

在着手设计项目之前，我们先上网查询了大量资料，并找到有经验的学长为我们答疑解惑。之后在项目修正测试阶段，由于实验室每周开放时间有限，不能满足我们的需求，我们借来了开发板和显示器，这样在宿舍也能继续我们的实验，测试时间富裕了很多。从五一假期开始到现在，我们花了很多个晚上在这个项目上，尽管很坎坷，但当改好最后一步，正好看到清晨四点半初升的太阳的时候，我们知道，这是最圆满的收尾。

4. HSF

1）Verilog 语言

第一次接触硬件描述语言，不同于 C、JAVA 等高级程设语言的顺序执行，verilog 语言更像 haskell，其代码编写顺序和执行顺序并无关系。函数的调用、模块的执行次序由模块间的信号决定。Verilog 编写时，尽量坚持一个.v 文件一个 module 原则。按照功能划分模块。文件的结构和整个系统的结构对应。尽管 verilog 语言和我最熟悉的 C 语言有很大区别，但不得不说还是很感谢 C 语言带我入门了与计算机对话的世界——无论硬件 or 软件、高级 or 底层。尤其因为 C 语言的设计也接近底层，让我对与硬件“对话”有一定的概念。而这次 verilog 和硬件的对话更可爱了，它的编写比汇编友善地多，却又很巧妙地操纵着硬件。

2）硬件设计与软件设计

软件设计更关注“流程”，即每一步要做什么，这一步下面要做什么，这一步的进行要提前做什么等等。而硬件更关注“信号处理”，这一模块对输入信号进行判断处理后，它的输出信号将作为另一个或者多个模块的输入，模块间工作独立，却又整体协作一个任务。所以软件的设计图为流程图，硬件则为状态转换图。由于绘图软件局限以及部分信号的展示不必要，最终提交了一张尽量完整的状态转换图。

3) 团队合作

这是一个矛盾，所谓众人拾柴火焰高，更多人的确能为团队贡献更多力量。但是个体的思想无法复制到另一个体大脑中，所以工作时也会出现不兼容的现象。所以初期设计很重要，需要更准确地设计好模块划分，确定接口，这样模块独立性更高、耦合度降低。

4) 谈点知识，更想谈谈情怀。

这次项目，我们小组前前后后花了一个月的时间。可以说每一决策都举步维艰，哪怕选题都是战战兢兢。因为之前的学习都是不明所以地依葫芦画瓢，基本按照流程走，便能得到一个成果，但是对于其中所以然我们不得而知。所以选题的时候很没有自信，太简单怕成绩不够理想，太难怕完不成。项目选题和初步设计由我和王紫薇决定，乐于挑战和勇于突破的我们最终还是决定做一个稍庞大的 VGA 项目，项目以经典游戏俄罗斯方块为载体。

所谓理想，在水一方。溯洄从之，道阻且长。每每利用空余时间为项目添砖加瓦或是修复润色之后满怀期待去实验室——无信号、黑屏、字模显示混乱……然后看着日子倒数着一天天变少，而进实验室实践测试的机会又那么稀缺，临近 deadline 的时候还有考试和繁重的作业，心态也越来越差。

但是秉着不懈的精神，我们还是坚持着项目的进程，每次进实验室发现一些 bug，当周修复这个问题，等待下一次实验室开放。周而复始。

像一篇小说，它交代了背景、人物、情节，它也为我们准备了高潮和结局——验收前夜——微机接口考试结束，直至次日凌晨 4 点半。借着实验室借来的开发板、显示器等，我们修复了残余的 bug，在第二天的展示中，也对自己的发挥比较满意。

最后想给课程提点建议：

1. 建议不要采纳有些同学削减小组成员的建议，因为大多同学的实践还是有很大提升空间的。即使按时完成任务的我们小组，也是熬夜通宵到当天凌晨 4 点多才做好的。
2. 建议大作业之前的授课对设计、原理等讲的更为详细；实践作业更改为稍带自我摸索形式的而非老师直接提供源码的硬件设计。
3. 建议实验室开放时间能分散一点，利于发现 bug 并预留时间用于修复。
4. 建议实验报告以组为单位而非以个人为单位，以留更多时间让同学进行动手操作。同时我认为共同完成报告利于小组成员的思维沟通。