可莉快跑

芝士雪豹队综合设计

中国科学技术大学

University of Science and Technology of China

报告人 徐航宇 闫泽轩 李牧龙

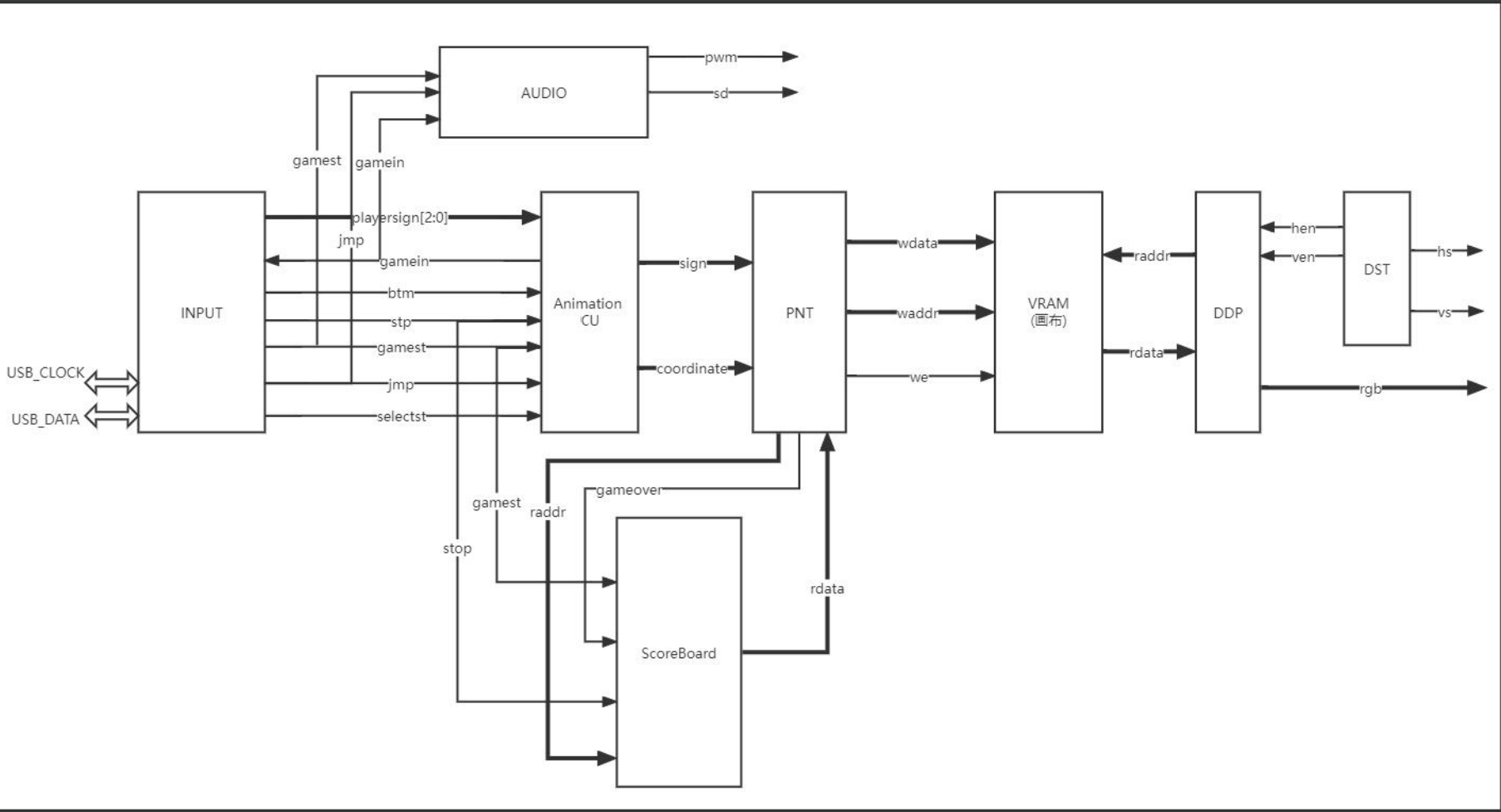
议程

- 视频展示
- 数据通路
- 分模块介绍
- 性能分析

视频展示

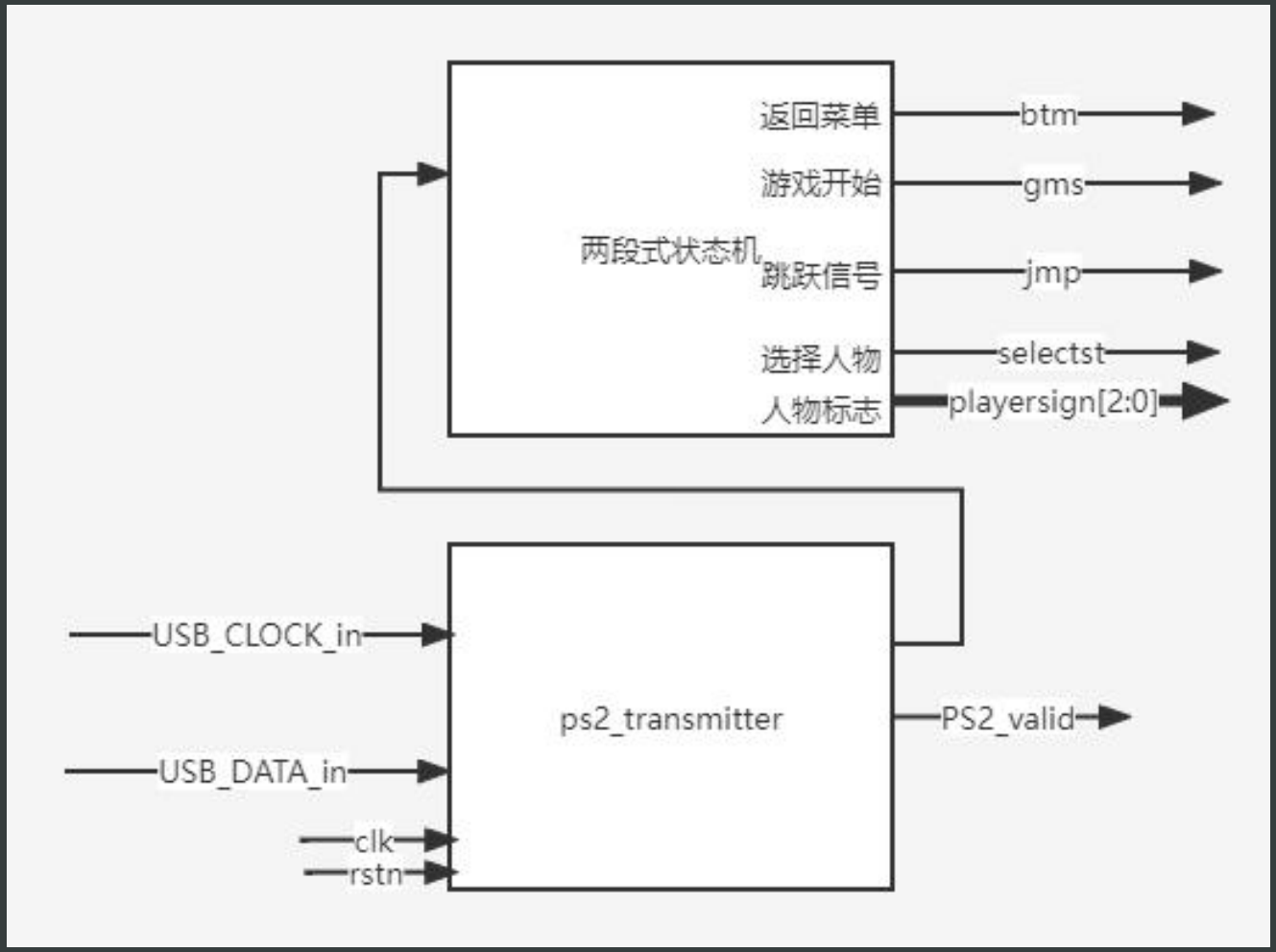
数据通路

顶层模块通路


AUDIO
pwm
sd
gamest
gamein
INPUT
playersign[2:0]
jmp
gamein
btm
stp
gamest
jmp
selectst
USB_CLOCK
USB_DATA
Animation
CU
sign
coordinate
PNT
wdata
waddr
we
VRAM
(画布)
raddr
rdata
DDP
hen
ven
DST
hs
vs
rgb
gamest
raddr
gameover
stop
rdata
ScoreBoard

分模块介绍

键盘输入模块


返回菜单
btm
游戏开始
gms
两段式状态机
跳跃信号
jmp
选择人物
selectst
人物标志
playersign[2:0]
USB_CLOCK_in
ps2_transmitter
PS2_valid
USB_DATA_in
clk
rstn

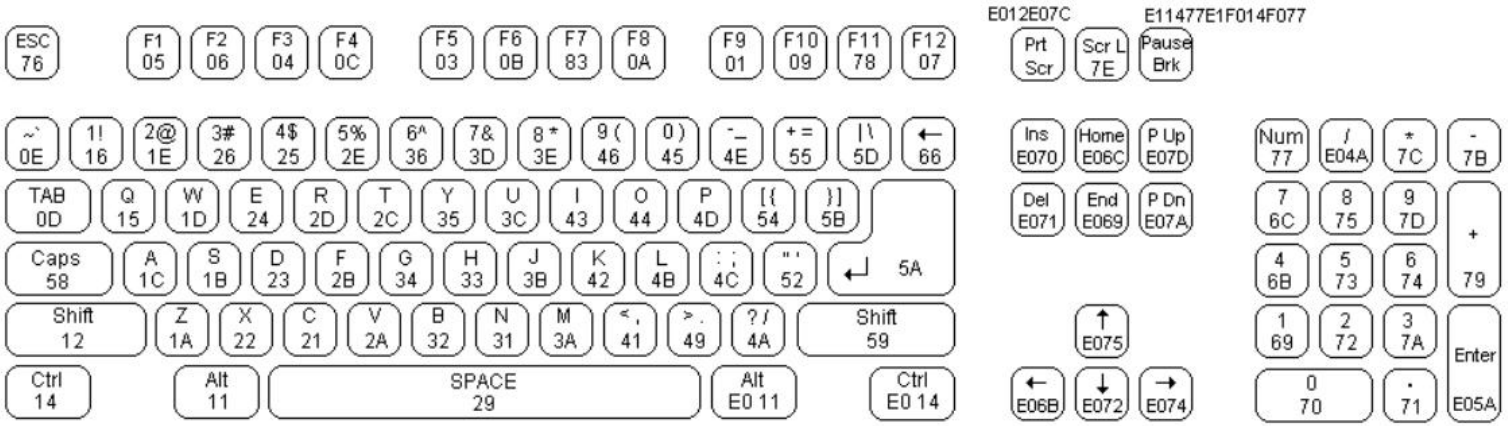
键盘信号编码

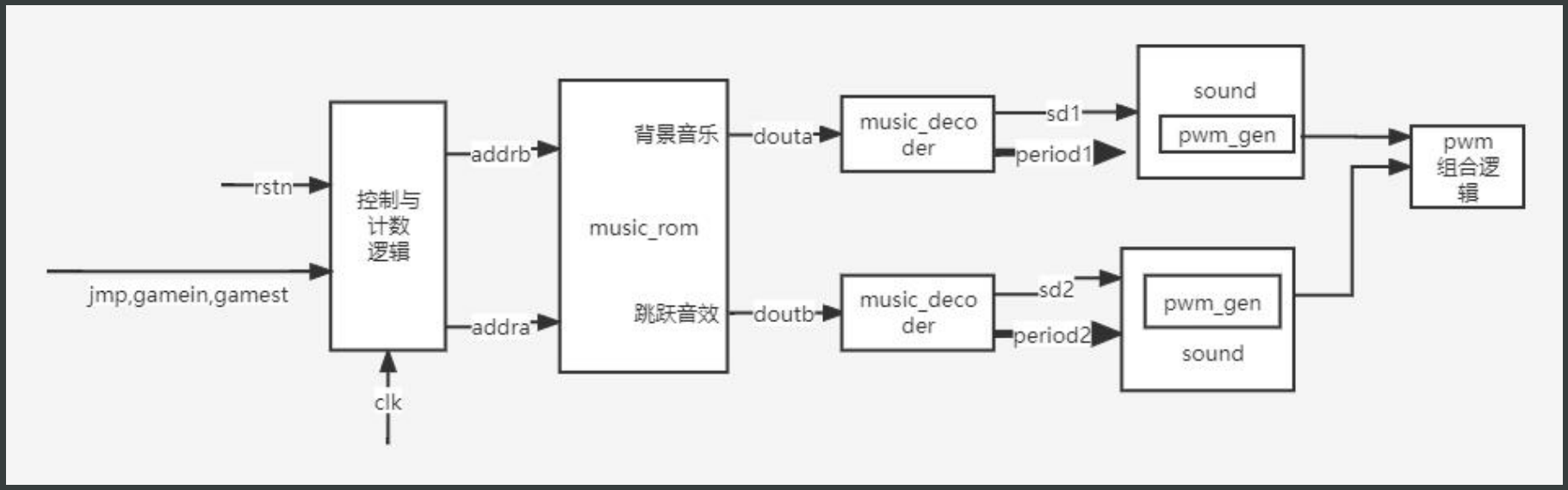
A. List of key codes

Graphic keys-to-keycodes map

Key names are on top, with the hexadecimal make code below.

Figure A.1. Keycode map




rstn
jmp,gamein,gamest
clk
控制与
计数
逻辑
addrb
addra
背景音乐
music_rom
跳跃音效
douta
doutb
music_deco
der
music_deco
der
sd1
period1
sd2
period2
sound
pwm_gen
pwm_gen
sound
pwm
组合逻
辑

AUDIO数据通路

音频输出原理

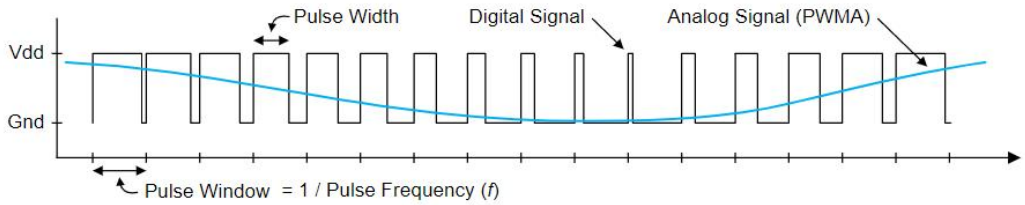


Figure 31. Simple waveform represented as PWM.

The PWM signal must be integrated to define an analog voltage. The low-pass filter 3dB frequency should be an order of magnitude lower than the PWM frequency, so that signal energy at the PWM frequency is filtered from the signal. For example, if an audio signal must contain up to 5 KHz of frequency information, then the PWM frequency should be at least 50 KHz (and preferably even higher). In general, in terms of analog signal fidelity, the higher the PWM frequency, the better. Figure 32 shows a representation of a PWM integrator producing an output voltage by integrating the pulse train. Note the steady-state filter output signal amplitude ratio to Vdd is the same as the pulse-width duty cycle (duty cycle is defined as pulse-high time divided by pulse-window time).

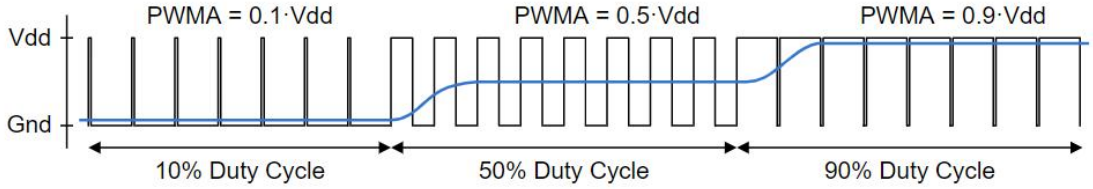


Figure 32. Representation of a PWM integrator producing an output voltage by integrating the pulse train.

频率编码处理

C调音符与频率对照表

音符	频率/Hz	音符	频率/Hz	音符	频率/Hz
60 低音1	262	72 中音1	523	84 高音1	1046
61 低音1#	277	73 中音1#	554	85 高音1#	1109
62 低音2	294	74 中音2	587	86 高音2	1175
63 低音2#	311	75 中音2#	622	87 高音2#	1245
64 低音3	330	76 中音3	659	88 高音3	1318
65 低音4	349	77 中音4	698	89 高音4	1397
66 低音4#	370	78 中音4#	740	90 高音4#	1480
67 低音5	392	79 中音5	784	91 高音5	1568
68 低音5#	415	80 中音5#	831	92 高音5#	1661
69 低音6	440	81 中音6	880	93 高音6	1760
70 低音6#	466	82 中音6#	932	94 高音6#	1865
71 低音7	494	83 中音7	988	95 高音7	1976

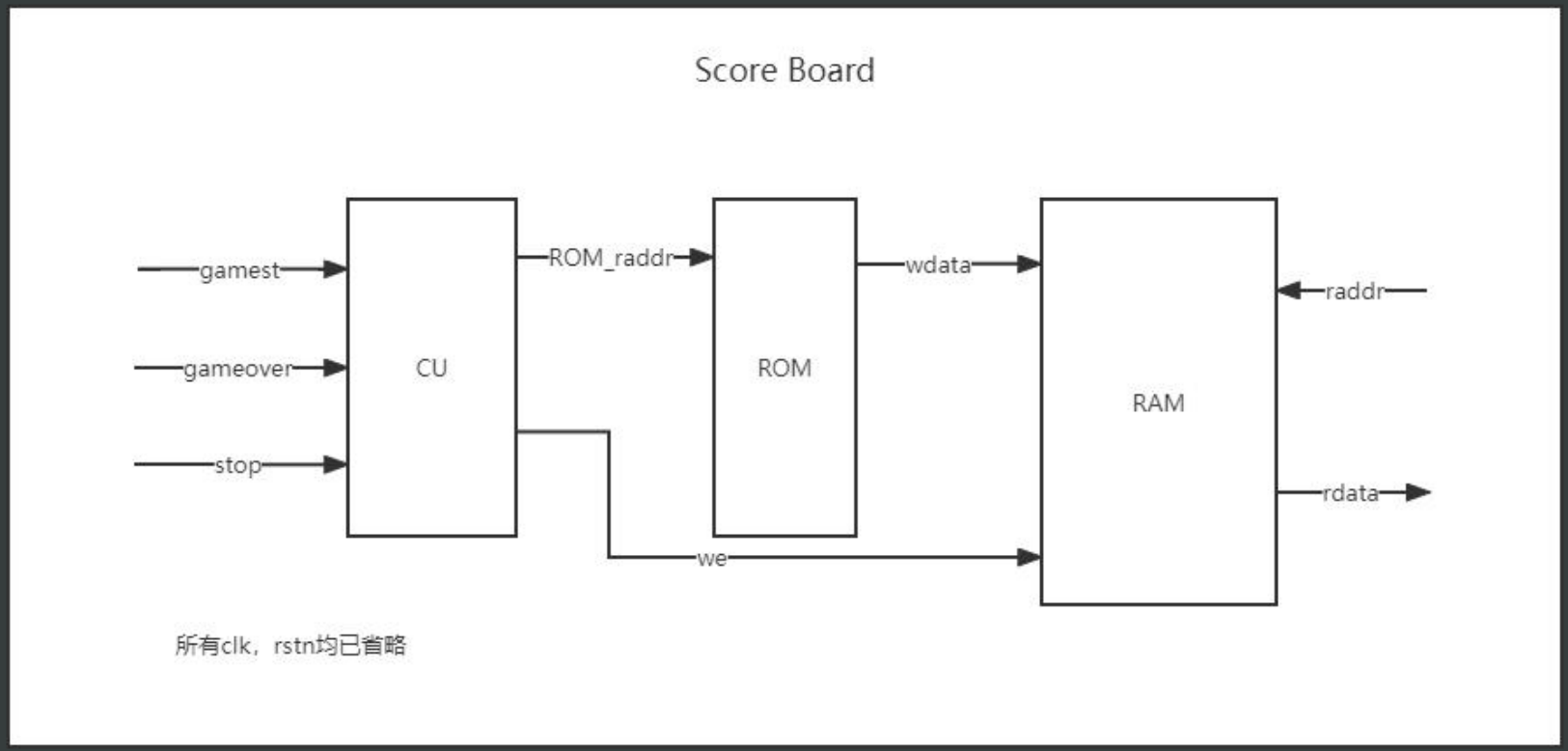
每两个半音频率之比为 $\sqrt[12]{2}$ ，

1#（升半音）和**2b**（降半音）的频率相同

```
rf1=[0, 0, 0, 17, 26, 26, 26, 25, 23, 23, 23, 22, 23, 23, 23, 23, 0, 0, 0, 27, 26, 26, 26, 25, 23, 23, 23, 22, 23, 23, 23, 23, 0, 0, 0, 27, 26, 26, 26,
    25, 23, 23, 23, 22,     23, 23, 23, 23,    25, 25, 25, 25,    26, 26, 26, 26,    27, 27, 27, 27]
rf2=[26, 26, 26, 26, 0, 0, 0, 27, 26, 26, 26, 25, 23, 23, 23, 22, 23, 23, 23, 23, 0, 0, 0, 27, 26, 26, 26, 25, 23, 23, 23, 22, 23, 23, 23, 23, 0, 0, 0, 23, 25, 25, 25, 26,
    27, 27, 27, 32,         33, 33, 33, 33,    32, 32, 32, 32,    33, 33, 33, 33,    35, 35, 35, 35]
rf3=[36, 36, 36, 36, 0, 0, 0, 37, 36, 36, 36, 35, 33, 33, 33, 32, 33, 33, 33, 33, 0, 0, 0, 37, 36, 36, 36, 35, 33, 33, 33, 32, 33, 33, 33, 33, 0, 0, 0,
    37, 36, 36, 36, 35, 33, 33, 33, 32, 33, 33, 33, 33,    35, 35, 35, 35,    36, 36, 36, 36,    37, 37, 37, 37]
rf4=[36, 36, 36, 36, 0, 0, 0, 37,    36, 36, 36, 35,    33, 33, 33, 32,    33, 33, 33, 33, 0, 0, 0, 37, 36, 36, 36, 35,    33, 33, 33, 32, 33, 33, 33, 33,    0, 0, 0, 33,
    35, 35, 35, 36,    37, 37, 37, 42,    43, 43, 43, 43, 42, 42, 42, 42,    43, 43, 43, 43,    45, 45, 45, 45]
rf5=[46, 46, 46, 46, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 22,
    23, 23, 23, 25,    23, 23, 23, 23, 23, 23, 23, 23,    17, 17, 17, 22,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 23, 23, 23, 22]
rf6=[23, 23, 23, 33, 32, 32, 32, 32,    27, 27, 27, 27,    26, 26, 26, 26,    27, 27, 27, 27,    27, 27, 27, 27,    27, 27, 27, 27,    27, 27, 27, 26,
    27, 27, 27, 32,    27, 27, 27, 27,    26, 26, 26, 26,    25, 25, 25, 26,    27, 27, 27, 27,    27, 27, 27, 27,    27, 27, 27, 27,    27, 27, 27, 26]
rf7=[27, 27, 27, 32,    27, 27, 27, 27,    26, 26, 26, 26,    25, 25, 25, 25,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 22,
    23, 23, 23, 25, 23, 23, 23, 23, 22, 22, 22, 22,    17, 17, 17, 22,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 23,    23, 23, 23, 22]
rf8=[23, 23, 23, 33, 32, 32, 32, 32,    32, 32, 32, 32, 27, 27, 27, 27, 26, 26, 26, 26,    27, 27, 27, 27, 27, 27, 27, 27,
    27, 27, 27, 26, 25, 25, 25, 26,    27, 27, 27, 27,    27, 27, 27, 27,    27, 27, 27, 26,    25, 25, 25, 26]
rf9=[27, 27, 27, 27,  27, 27, 27, 27,  27, 27, 27, 27,  27, 27, 27, 27,  22, 22, 22, 22,  22, 22, 22, 22,  22, 22, 22, 22,  22, 22, 22, 22]
```

```
In [1]:
dic={11:"00001",12:"00010"}
dic[13]="00011"
dic[14]="00100"
dic[15]="00101"
dic[16]="00110"
dic[17]="00111"
dic[21]="01000"
dic[22]="01001"
dic[23]="01010"
dic[22]="01001"
dic[24]="01011"
dic[25]="01100"
dic[26]="01101"
dic[27]="01110"
dic[31]="01111"
dic[32]="10000"
dic[33]="10001"
dic[34]="10010"
dic[35]="10011"
dic[36]="10100"
dic[37]="10101"
dic[38]="10110"
dic[39]="10111"
dic[41]="11000"
dic[42]="11001"
dic[43]="11010"
dic[44]="11011"
dic[45]="11100"
dic[46]="11101"
dic[47]="11110"
dic[0]="00000"
```

计分板模块


Score Board
gamest
gameover
stop
CU
ROM_raddr
ROM
wdata
RAM
raddr
rdata
we
所有clk，rstn均已省略

随机数生成器

- 原理：梅森旋转
- 使用32位移位寄存器，使用前输入种子进行初始化，随后每次生成随机数时都取出寄存器中的几位，进行异或运算后存入低位。
- 使用时截取其中几位即可

动画控制与显示(设计模式)

共画布

所有动画显示都读取自一个VRAM，我们称之为画布。每一帧都对画布进行刷新，即对VRAM进行写操作
这一设计本质上分离了实际的显示模块与动画设置模块

对象分离

将各个需要显示的动画拆分为若干对象，并将信息分别存储于各ROM中。每一帧根据对象的sign和coordinate将动画信息写到画布VRAM上
这一设计本质上分离了画布的刷写与动画的控制

动画控制与显示(设计模式)

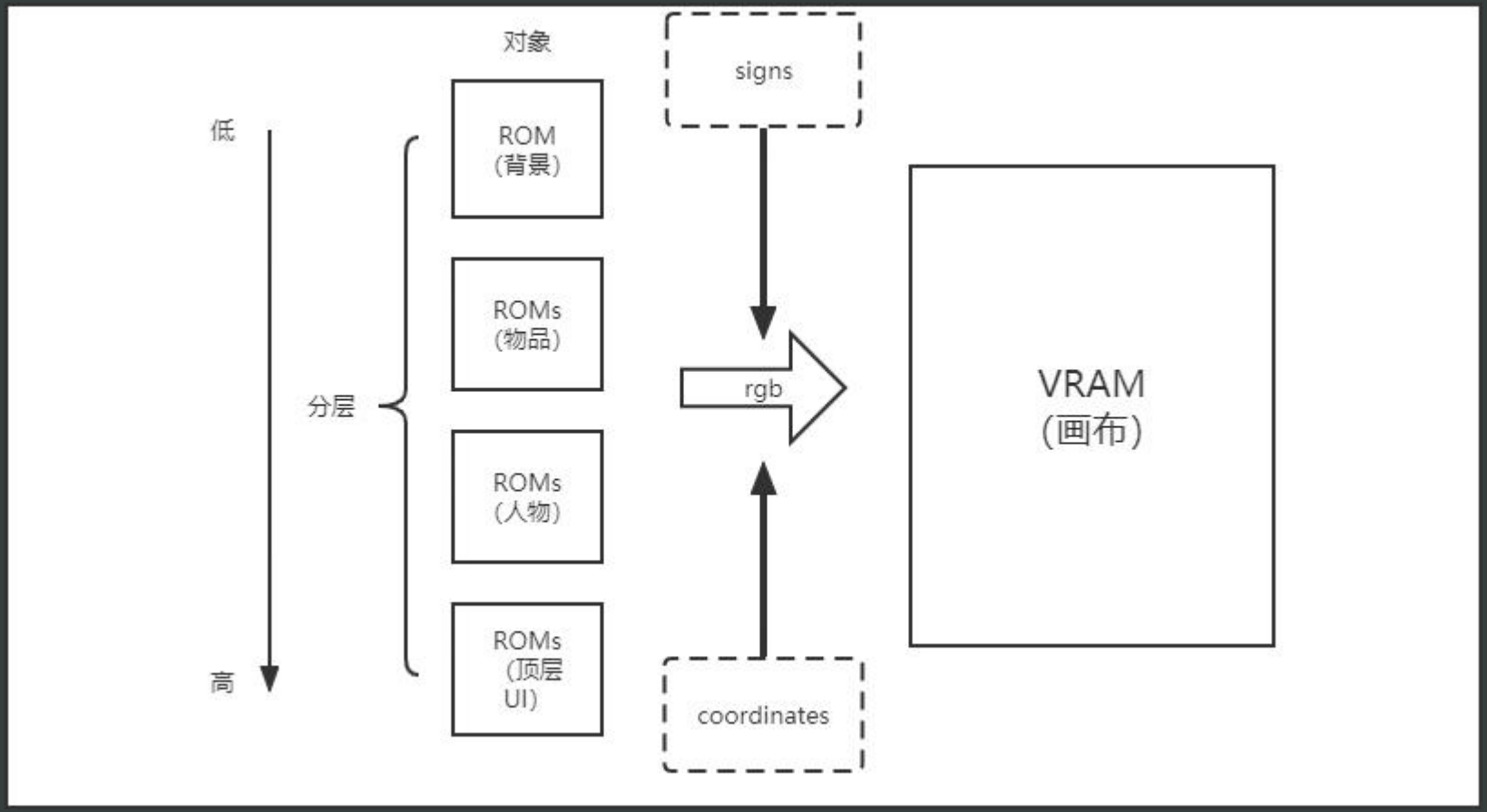
对象分层

各对象根据显示的优先级分层，可能发生冲突的对象置于不同层中，并按层级显示其中一层的信息。
可在指定的两层发生冲突时，发出反馈信号，以此实现碰撞反馈等机制
各层有分离的rdata和raddr接线，为透明度的实现提供便利

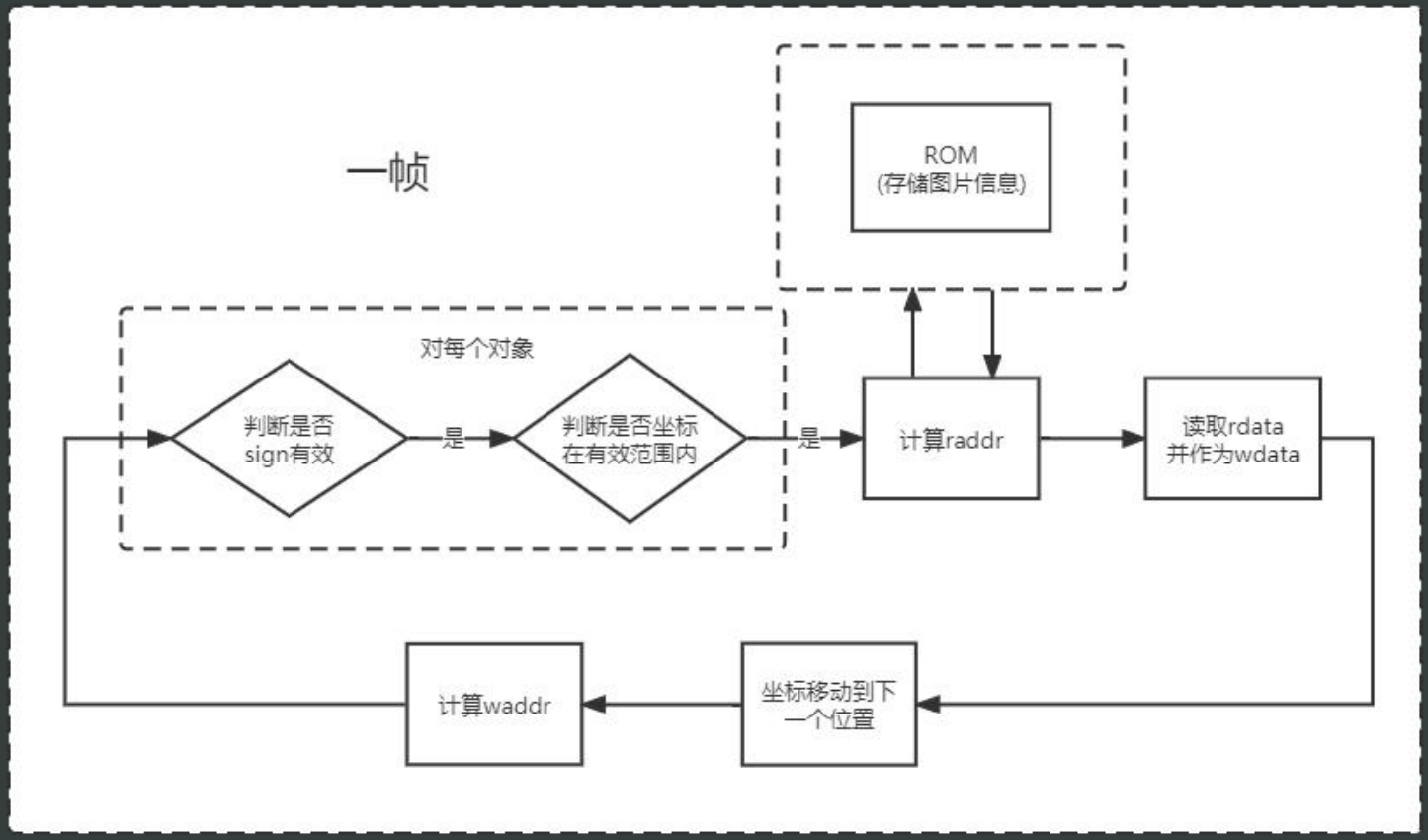
面向对象

将动画的显示，拆分为对对象的位置及是否显示等属性的修改，并交由动画控制模块进行控制（例子：跳跃、跑动）
这一设计本质上分离了动画与位移

动画控制与显示(设计模式)



动画控制与显示(具体流程)



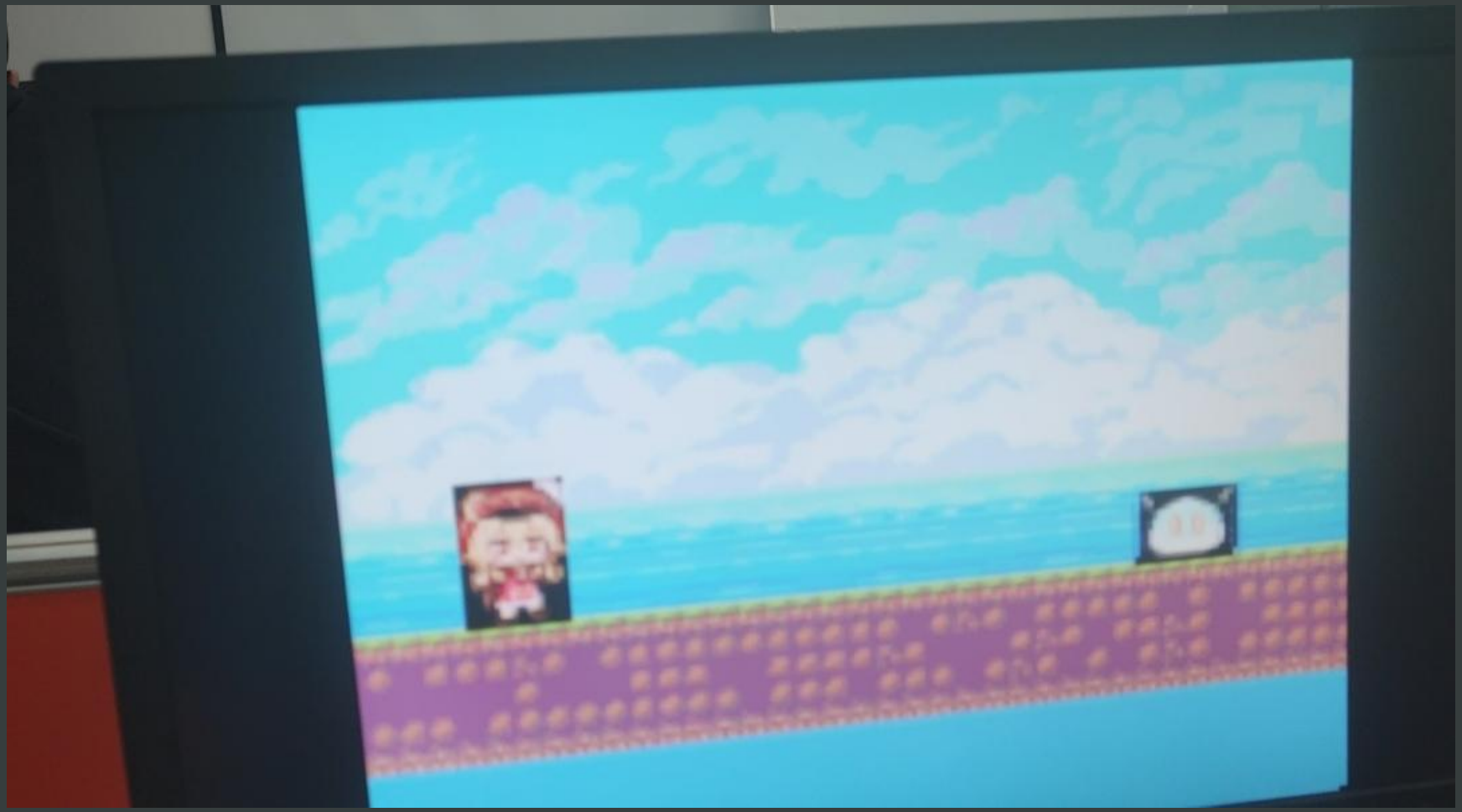
动画控制与显示(各功能具体实现)

透明

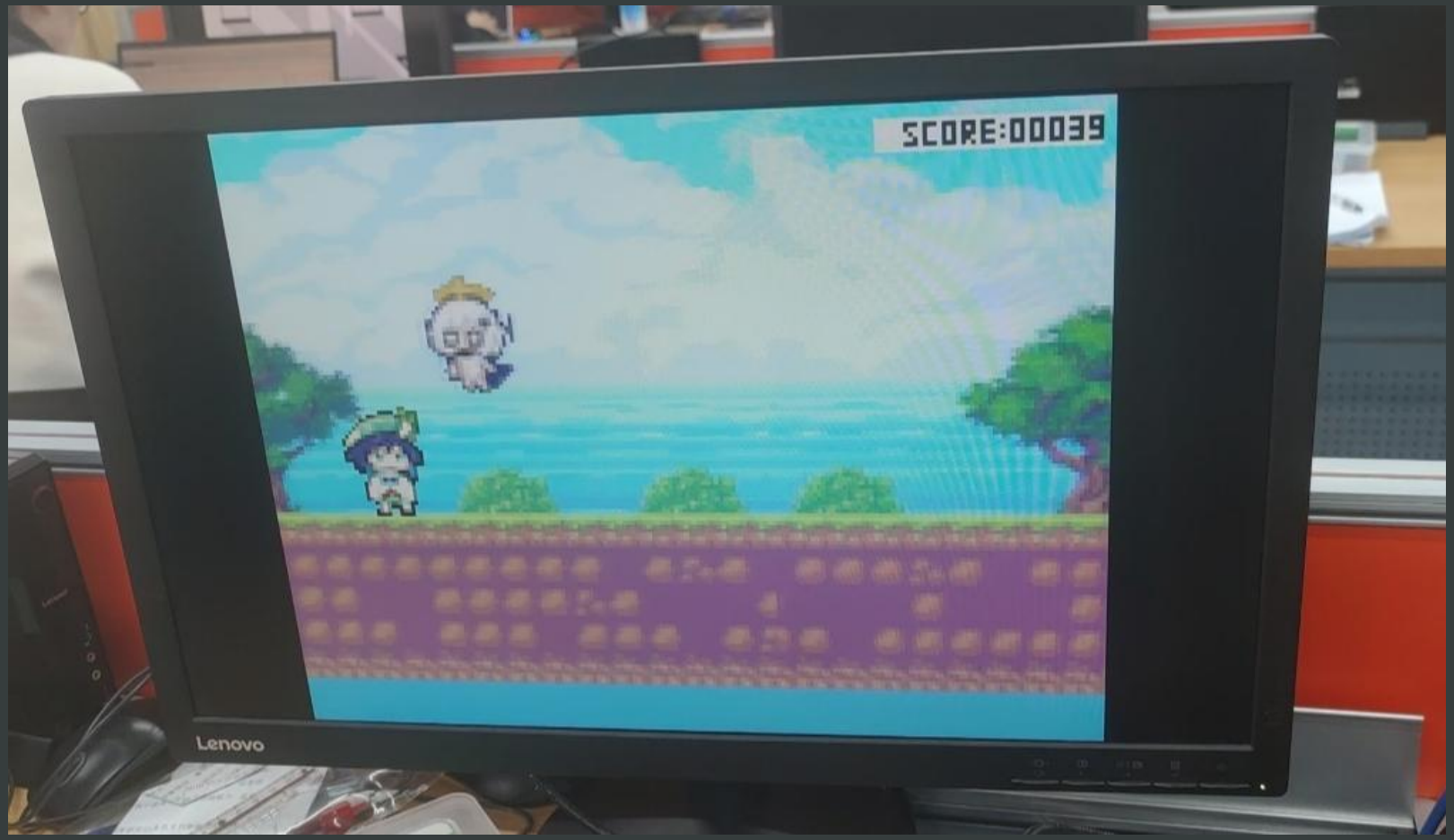
对画布上的某一点，当检测到待输出的rdata中alpha=1，转而输出下一层中的对应rdata

```
if(dout_p0_run0[3:0]==4'b000) wdata<=dout_bk;
else wdata<=dout_p0_run0[15:4];
```

动画控制与显示(各功能具体实现)



动画控制与显示(各功能具体实现)



动画控制与显示(各功能具体实现)

碰撞反馈

当在刷写画布上某一点时，怪物和人物的sign同时有效，且同时在有效坐标范围内，判定为发生碰撞，发出反馈，并停止游戏

```
if(x_coordinate>=p1_x_coordinate
&& x_coordinate<=p2_x_coordinate
&& y_coordinate>=p1_y_coordinate
&& y_coordinate<=p2_y_coordinate
) begin
    //碰撞
    if(sm_sign<4'd12
    && x_coordinate>=sm1_x_coordinate
    && x_coordinate<=sm2_x_coordinate
    && y_coordinate>=sm1_y_coordinate
    && y_coordinate<=sm2_y_coordinate
    ) begin
        gameover<=1;
    end
```

动画控制与显示(各功能具体实现)

人物选择	通过来自输入的player_sign切换
人物走动	在游戏进行中，每隔RunTime时间，对run_sign自增
人物跳动	在接收到跳跃信号后，每隔JumpTime时间，对player的y_coordinate进行修改

动画控制与显示(各功能具体实现)

怪物刷新	每当怪物的x1_coordinate(左侧)为0时，通过随机数发生器对sm_sign赋值；
怪物移动	在游戏进行中，每隔RollTime时间，对怪物x_coordinate进行更新 由于存在空中单位和地面单位，根据怪物类型设置其y_coordinate
背景滚动	在游戏进行中，每隔RollTime时间，对背景读取的起始坐标x_coordinate进行循环递增

性能分析

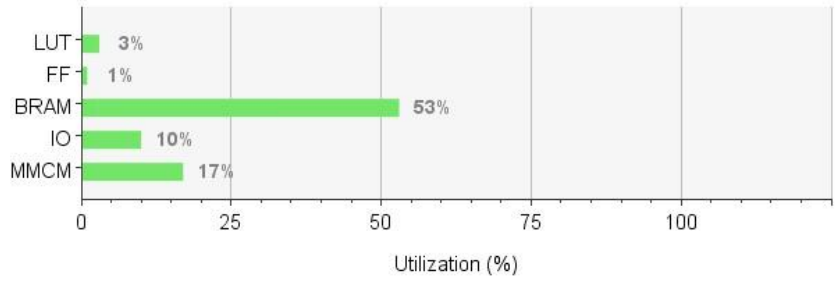
资源使用量

Tcl Console | Messages | Log | Reports | Design Runs | DRC | Methodology | Power | Timing | **Utilization**

Summary

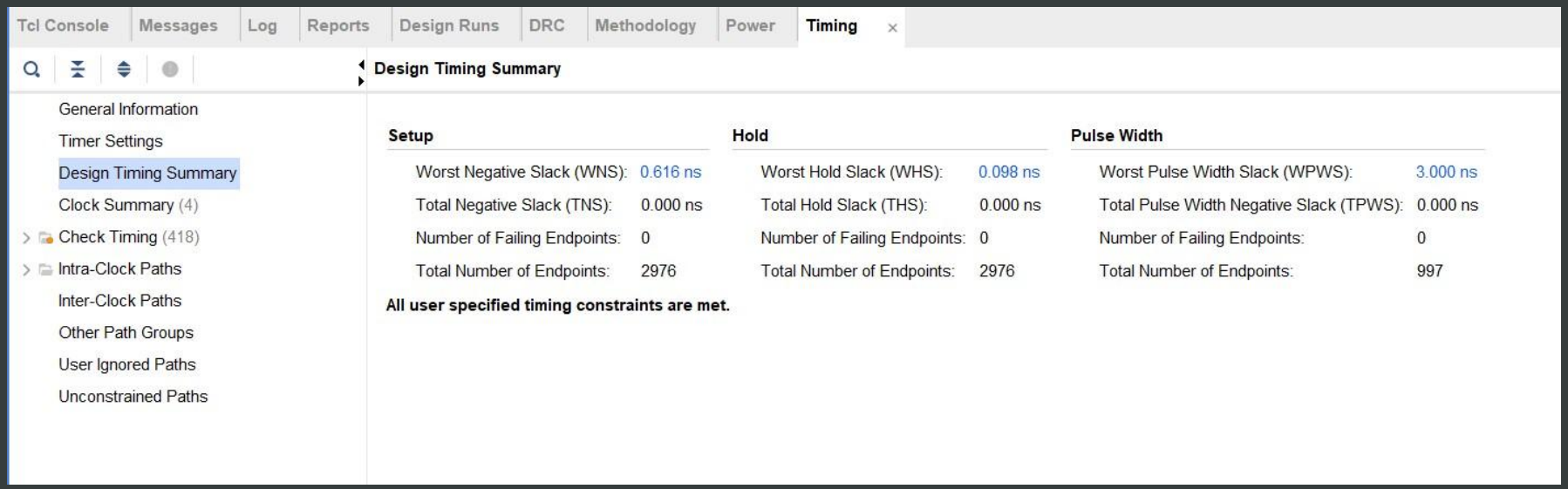
- Hierarchy
- Summary
- Slice Logic
  - F7 Muxes (<1%)
  - Slice Registers (1%)
    - Register as Flip Flop (1
    - Register as Latch (<1%
  - Slice LUTs (3%)
    - LUT as Logic (3%)
- Slice Logic Distribution
  - Slice Registers (1%)
    - Register driven from wi
    - Register driven from ou
      - LUT in front of the r
      - LUT in front of the r
  - LUT as Logic (3%)
    - using O5 and O6<>LUT
    - using O6 output only<>

Resource	Utilization	Available	Utilization %
LUT	2157	63400	3.40
FF	896	126800	0.71
BRAM	71	135	52.59
IO	20	210	9.52
MMCM	1	6	16.67



时序情况

时钟频率为100MHz



创寰宇学府 育天下英才

感谢观看

中国科学技术大学

University of Science and Technology of China

报告人 徐航宇 闫泽轩 李牧龙