**音乐播放模块**

**Music.v**

module music(

input clk,

input stop,//暂停

input [9:0] timeunit,

output reg speaker,

input start//复位

);

//分频计数器

parameter wide=15;

reg[7:0] cnt; //音名数

reg[4:0] cnt1; //20MHz基频

reg[27:0] cnt2;//节拍频率

reg[wide-1:0] origin;//预置数寄存器

reg[wide-1:0] drive;

reg[1:0] count;

reg carrier;

reg [27:0] freq;

always@(posedge clk)

freq<=100000\*timeunit;

always @(posedge clk or posedge start)

begin

if(start)

begin

cnt1<=5'd0;

cnt2<=27'd0;

end

else if(~stop)

begin

cnt1<=cnt1+1'b1;

cnt2<=cnt2+1'b1;

if(cnt1==5'd19)

cnt1<=5'd0;

if(cnt2==freq)//10^7

cnt2<=27'd0;

end

end

always @(posedge clk or posedge start)

begin

if(start)

drive<=15'h0;

else if(~stop)

begin

if(cnt1==5'd19)

begin

if(drive==15'd32767)

begin

drive<=origin;

carrier<=1'b1;

end

else

begin

drive<=drive+1'b1;

carrier<=1'b0;

end

end

end

end

//carrier的频率是每个音阶的频率

always @(posedge carrier)

begin

count<=count+1'b1;

if(count==4'd0)//16

speaker<=1'b1;

else speaker<=1'b0;

end

always @(posedge clk or posedge start)

begin

if(start)

begin

origin<=15'h0;

cnt<=8'd0;

end

else if(~stop && cnt2==freq)

begin

if(cnt==8'd70)

cnt<=8'd0;

else

cnt<=cnt+1'b1;

case (cnt)

8'd0:origin<=15'd32767;

8'd1:origin<=15'd32767;

8'd2:origin<=15'd27085;

8'd3:origin<=15'd27085;

8'd4:origin<=15'd27085;

8'd5:origin<=15'd26389;

8'd6:origin<=15'd25191;

8'd7:origin<=15'd25191;

8'd8:origin<=15'd26389;

8'd9:origin<=15'd26389;

8'd10:origin<=15'd27987;

8'd11:origin<=15'd27987;

8'd12:origin<=15'd27085;

8'd13:origin<=15'd26389;

8'd14:origin<=15'd27085;

8'd15:origin<=15'd27085;

8'd16:origin<=15'd27085;

8'd17:origin<=15'd27085;

8'd18:origin<=15'd25191;

8'd19:origin<=15'd25191;

8'd20:origin<=15'd26389;

8'd21:origin<=15'd27085;

8'd22:origin<=15'd26389;

8'd23:origin<=15'd26389;

8'd24:origin<=15'd25191;

8'd25:origin<=15'd25191;

8'd26:origin<=15'd23225;

8'd27:origin<=15'd21403;

8'd28:origin<=15'd26389;

8'd29:origin<=15'd25191;

8'd30:origin<=15'd24264;

8'd31:origin<=15'd24264;

8'd32:origin<=15'd24264;

8'd33:origin<=15'd24264;

8'd34:origin<=15'd24264;

8'd35:origin<=15'd24264;

8'd36:origin<=15'd24264;

8'd37:origin<=15'd25191;

8'd38:origin<=15'd26389;

8'd39:origin<=15'd26389;

8'd40:origin<=15'd26389;

8'd41:origin<=15'd27402;

8'd42:origin<=15'd25191;

8'd43:origin<=15'd25191;

8'd44:origin<=15'd24264;

8'd45:origin<=15'd24264;

8'd46:origin<=15'd23225;

8'd47:origin<=15'd23225;

8'd48:origin<=15'd23225;

8'd49:origin<=15'd23225;

8'd50:origin<=15'd26389;

8'd51:origin<=15'd26389;

8'd52:origin<=15'd26389;

8'd53:origin<=15'd25191;

8'd54:origin<=15'd24264;

8'd55:origin<=15'd23225;

8'd56:origin<=15'd21403;

8'd57:origin<=15'd23225;

8'd58:origin<=15'd20012;

8'd59:origin<=15'd20012;

8'd60:origin<=15'd20012;

8'd61:origin<=15'd20012;

8'd62:origin<=15'd20012;

8'd63:origin<=15'd20012;

8'd64:origin<=15'd20012;

8'd65:origin<=15'd20012;

default:origin<=15'd32767;

endcase

end

end

endmodule

**难度选择模块difficultysel.v**

module difficultysel(

input clk,

input start,//记得最好给所有模块上一个复位信号

input [2:0] difficulty,

output reg [9:0] timeunit

);

always@(posedge clk or posedge start)

if(start)

timeunit=300;

else

case(difficulty)//由于数码管的限制，我无法改变键位的密度，只能通过曲子节奏的快慢来改变难度

3'b001:timeunit=700;

3'b010:timeunit=500;

3'b100:timeunit=300;

default: timeunit=500;//!!!需要更改！

endcase

endmodule

数码管扫描信号生成模块：

**Scan.v**

module scan(

input clk,

input start,

output reg scan\_sgn

);

reg [16:0] scan\_cnt;

always@(posedge clk or posedge start)

begin

if(start)

scan\_cnt<=0;

else if(scan\_cnt==17'd9\_999)

scan\_cnt<=17'd0;

else

scan\_cnt<=scan\_cnt+17'b1;

end

always@(posedge clk or posedge start)

begin

if(start)

scan\_sgn<=0;

else if(scan\_cnt==17'd9\_999)

scan\_sgn<=1'b1;

else

scan\_sgn<=1'b0;

end

endmodule

**判定模块：**

**Judge.v**

module judge(

input btnu,

input stop,

input btnd,

input btnl,

input btnr,

input start,

input clk,

input [7:0] Y,

input OE,

input [9:0] timeunit,

output reg [2:0] level,

output reg [4:0] rank

);

reg [12:0] score;

reg [16:0] counter;//(10^5)

reg [18:0] cntms;//不超过1min(60000ms)的游戏

reg playing;

reg [2:0] state [3:0];

reg [0:0] turn1 [1:0];//判断长键信号是否为拐弯信号，0表示非,共两个1位

reg [18:0] queue1 [5:0];//存储按键发生时间和长度

reg [18:0] queue2 [5:0]; //由于for的限制，不得已将queue延长

reg [18:0] queue3 [5:0];

reg [0:0] turn4 [1:0];//二维数组先位宽再位深！

reg [18:0] queue4 [5:0];//长

integer k;

reg [18:0] tdist [4:0];//记录与标准时间的差值

reg [18:0] sttime1;

reg [18:0] sttime4;

wire btnu\_pos\_p;

wire btnd\_pos\_p;

wire btnl\_pos\_p;

wire btnr\_pos\_p;

wire btnu\_neg\_p;

wire btnd\_neg\_p;

wire btnl\_neg\_p;

wire btnr\_neg\_p;

reg btnu\_pos\_r;

reg btnu\_neg\_r;

reg btnd\_pos\_r;

reg btnd\_neg\_r;

reg btnl\_pos\_r;

reg btnl\_neg\_r;

reg btnr\_pos\_r;

reg btnr\_neg\_r;

signal\_generate u\_signal\_generate(

.reset(start),

.signal\_in(btnu),

.clk(clk),

.signal\_out\_pos(btnu\_pos\_p),

.signal\_out\_neg(btnu\_neg\_p)

);

signal\_generate d\_signal\_generate(

.reset(start),

.signal\_in(btnd),

.clk(clk),

.signal\_out\_pos(btnd\_pos\_p),

.signal\_out\_neg(btnd\_neg\_p)

);

signal\_generate l\_signal\_generate(

.reset(start),

.signal\_in(btnl),

.clk(clk),

.signal\_out\_pos(btnl\_pos\_p),

.signal\_out\_neg(btnl\_neg\_p)

);

signal\_generate r\_signal\_generate(

.reset(start),

.signal\_in(btnr),

.clk(clk),

.signal\_out\_pos(btnr\_pos\_p),

.signal\_out\_neg(btnr\_neg\_p)

);

always@(posedge clk)

begin

btnu\_pos\_r=btnu\_pos\_p;

btnu\_neg\_r=btnu\_neg\_p;

btnd\_pos\_r=btnd\_pos\_p;

btnd\_neg\_r=btnd\_neg\_p;

btnl\_pos\_r=btnl\_pos\_p;

btnl\_neg\_r=btnl\_neg\_p;

btnr\_pos\_r=btnr\_pos\_p;

btnr\_neg\_r=btnr\_neg\_p;

end

always@(posedge clk or posedge start)//系统时间

if(start)

begin

counter=0;

playing=1;

cntms=0;

end

else if(~stop)

begin

if(playing)

if(counter==17'd99999)

begin

counter=0;

cntms=cntms+1;

end

else

counter=counter+1;

end

always@(posedge clk or posedge start)

begin

if(start)

begin

level<=3'b0;

for(k=0;k<5;k=k+1)//注意二维数组必须这样初始化

begin

queue1[k]<=0;

queue2[k]<=0;

queue3[k]<=0;

queue4[k]<=0;

tdist[k]<=0;

end

for(k=0;k<2;k=k+1)

begin

turn1[k]<=0;

turn4[k]<=0;

end

score<=0;

sttime1<=0; //长键的按下去的时间

sttime4<=0;

end

else if(~stop)//\*\*\*

begin

//\*\*\*\*\*\*\*\*\*左长键\*\*\*\*\*\*\*\*\*

if(btnl\_pos\_r)

begin

state[0]<=3'd1;

sttime1<=cntms;

end

else if(btnl\_neg\_r)

begin

state[0]<=3'd2;

end

case(state[0])

3'd0:

begin

end

3'd1://记录开始发生时间

begin

state[0]<=3'd0;

end

3'd2:

begin

if(sttime1>queue1[0])//miss

state[0]<=3'd3;

else

begin

if(cntms<queue1[0]+queue1[1]\*timeunit)//提前结束按键

if(cntms>queue1[0])

begin

tdist[0]<=cntms-queue1[0];

state[0]<=3'd4;//去结算

end

else

state[0]<=3'd6;

else

begin

tdist[0]<=queue1[1]\*timeunit;

state[0]<=3'd4;//去结算

end

end

end

3'd3://miss

begin

level<=3'b001;

state[0]<=3'd5;

end

3'd4://长键结算

begin

score<=score+(tdist[0]/timeunit)\*50;

state[0]<=3'd5;

level<=3'b111;//perfect

end

3'd5://队列变化

begin

queue1[0]<=queue1[2];

queue1[1]<=queue1[3];

queue1[2]<=0;

queue1[3]<=0;

queue1[4]<=0;

state[0]<=3'd6;

end

3'd6:

begin

tdist[0]<=0;

sttime1<=0;

state[0]<=3'd0;

end

endcase

//\*\*\*\*\*\*\*\*\*\*\*\*右长键\*\*\*\*\*\*\*\*\*\*\*\*

if(btnu\_pos\_r)

begin

state[3]<=3'd1;

sttime4<=cntms;

end

else if(btnu\_neg\_r)

begin

state[3]<=3'd2;

end

case(state[3])

3'd0:

begin

end

3'd1://记录开始发生时间和miss

begin

state[3]<=0;

end

3'd2:

begin

if(sttime4>queue4[0])//miss

state[3]<=3'd3;

else

begin

if(cntms<queue4[0]+queue4[1]\*timeunit)//提前结束按键

if(cntms>queue4[0])

begin

tdist[3]<=cntms-queue4[0];//注意！！！这里必须保证和为正，否则一旦溢出，直接爆表

state[3]<=3'd4;//去结算

end

else

state[3]<=3'd6;

else

begin

tdist[3]<=queue4[1]\*timeunit;

state[3]<=3'd4;//去结算

end

end

end

3'd3://miss

begin

level<=3'b001;

state[3]<=3'd5;

end

3'd4://长键结算

begin

score<=score+(tdist[3]/timeunit)\*50;

state[3]<=3'd5;

level<=3'b111;//perfect

end

3'd5://队列变化

begin

queue4[0]<=queue4[2];

queue4[1]<=queue4[3];

queue4[2]<=0;

queue4[3]<=0;

queue4[4]<=0;

state[3]<=3'd6;

end

3'd6:

begin

tdist[3]<=0;

sttime4<=0;

state[3]<=0;

end

endcase

//\*\*\*\*\*\*\*\*\*\*右短键\*\*\*\*\*\*\*\*\*\*\*

if(btnr\_pos\_r)

begin

if(btnu && turn4[0])

state[2]<=3'd1;

else

state[2]<=3'd2;

end

case(state[2])

3'd0:

begin//这一句不加会报错

end

3'd1://清除拐弯标记

begin

score<=score+50;

turn4[0]<=turn4[1];

turn4[1]<=0;

state[2]<=3'd0;//直接结束

end

3'd2://计算误差

begin

if(cntms>queue3[0])

state[2]<=3'd3;//miss

else

begin

tdist[2]<=(queue3[0]-cntms);

state[2]<=3'd4;

end

end

3'd3://miss

begin

level<=3'b001;

state[2]<=3'd5; //清队列

end

3'd4://判断并给出得分

begin

if(tdist[2]<100)

begin

score<=score+13'd50;//perfect

level<=3'b111;

state[2]<=3'd5;

end

else if(tdist[2]<3\*timeunit)

begin

score<=score+13'd25;//great

level<=3'b011;

state[2]<=3'd5;

end

end

3'd5://清队列

begin

queue3[0]<=queue3[1];

queue3[1]<=queue3[2];

queue3[2]<=queue3[3];

queue3[3]<=0;

state[2]<=3'd6;

end

3'd6:

begin

tdist[2]<=0;

state[2]<=3'd0;

end

endcase

//\*\*\*\*\*\*\*\*\*\*\*左短键\*\*\*\*\*\*\*\*\*\*\*\*

if(btnd\_pos\_r)

begin

if(btnl && turn1[0])

state[1]<=3'd1;

else

state[1]<=3'd2;

end

case(state[1])

3'd0:

begin

end

3'd1:

begin

score<=score+50;

turn1[0]<=turn1[1];

turn1[1]<=0;

state[1]<=3'd0;//直接结束

end

3'd2://计算误差

begin

if(cntms>queue2[0])

state[1]<=3'd3;

else

begin

tdist[1]<=(queue2[0]-cntms);

state[1]<=3'd4;

end

end

3'd3://miss

begin

level<=3'b001;

state[1]<=3'd5; //清队列

end

3'd4://判断并给出得分

begin

if(tdist[1]<100)

begin

score<=score+13'd50;//perfect

level<=3'b111;

state[1]<=3'd5;

end

else if(tdist[1]<3\*timeunit)

begin

score<=score+13'd25;//great

level<=3'b011;

state[1]<=3'd5;

end

end

3'd5://清队列

begin

queue2[0]<=queue2[1];

queue2[1]<=queue2[2];

queue2[2]<=queue2[3];

queue2[3]<=0;

state[1]<=3'd6;

end

3'd6:

begin

tdist[1]<=0;

state[1]<=3'd0;

end

endcase

if(OE)//插入队列

begin

if(~Y[5])//左短键

begin

for(k=0;k!=4 && queue2[k]!=0;k=k+1)

begin

end

queue2[k]<=cntms+3\*timeunit;

end

if(~Y[3])//拐弯键

begin

if(turn1[0])

turn1[1]<=1;

else

turn1[0]<=1;

end

if(~Y[4])//左长键

begin

if(queue1[4]==0)

begin

if(queue1[0]==0)

begin

queue1[0]<=cntms+3\*timeunit;

end

queue1[1]<=queue1[1]+1;

end

else

begin

if(queue1[2]==0)

begin

queue1[2]<=cntms+3\*timeunit;

end

queue1[3]<=queue1[3]+1;

end

end

else

begin

if(queue1[4]==0 && queue1[0]!=0)

queue1[4]<=1;

end

if(~Y[1])//拐弯键

begin

if(turn4[0])

turn4[1]<=1;

else

turn4[0]<=1;

end

if(~Y[7])//右长键

begin

if(queue4[4]==0)

begin

if(queue4[0]==0)

begin

queue4[0]<=cntms+3\*timeunit;

end

queue4[1]<=queue4[1]+1;

end

else

begin

if(queue4[2]==0)

begin

queue4[2]<=cntms+3\*timeunit;

end

queue4[3]<=queue4[3]+1;

end

end

else

begin

if(queue4[4]==0 && queue4[0]!=0)

queue4[4]<=1;

end

if(~Y[6])//右短键

begin

for(k=0;k!=4 && queue3[k]!=0;k=k+1)

begin

end

queue3[k]<=cntms+3\*timeunit;

end

end//OE end

end//else end

end//always end

always@(score)

if(start)

rank=0;

else

begin

if(score>13'd0)

begin

if(score<13'd100)

rank=5'b00001;

else if(score<13'd300)

rank=5'b00011;

else if(score<13'd1000)

rank=5'b00111;

else if(score<13'd2000)

rank=5'b01111;

else// if(score<13'd5000)

rank=5'b11111;

end

else

rank=5'b00000;

end

endmodule

**键位数码管信号生成模块：**

**Mom.v**

module mom(

input clk,

input stop,

input start,

output reg [7:0] Y,

output reg OE,

input [9:0] timeunit

);

reg [16:0] counter;

reg [18:0] cntms;

reg playing;//如果将它定义成1，就会变成锁存器

reg [6:0] beat;

reg OElast;

always@(posedge clk or posedge start)

begin

if(start)

begin

OE<=0;

OElast<=0;

counter<=17'd0;

cntms<=16'd0;

playing<=1;

beat<=0;

Y=8'b1111\_1111;

//经查阅，必须在各个状态下都要给所有寄存器赋值，否则就会出现什么latch的错误

end

else if(~stop)

begin

if(playing && beat<7'd100)

begin

if(counter==17'd99\_999)

begin

OE<=OElast;

counter<=17'd0;

if(cntms==timeunit)

begin

cntms<=0;

beat<=beat+1;

OElast<=1;

end

else

begin

if(cntms<19'd256\_000)//cntms的上限足够大，不清零

cntms<=cntms+1;

end

end

else

begin

counter<=counter+1;

OElast<=0;

OE<=OElast;

end

end

if(playing && OElast)

case(beat)

7'd0:Y=8'b01111111;

7'd1:Y=8'b01111111;

7'd2:Y=8'b01111111;

7'd3:Y=8'b10111111;

7'd4:Y=8'b11011111;

7'd5:Y=8'b1111\_1111;

7'd6:Y=8'b10111111;

7'd7:Y=8'b1111\_1111;

7'd8:Y=8'b01111111;

7'd9:Y=8'b0111\_1111;

7'd10:Y=8'b10111111;

7'd11:Y=8'b11011111;

7'd12:Y=8'b01111111;

7'd13:Y=8'b01111111;

7'd14:Y=8'b01111111;

7'd15:Y=8'b01111111;

7'd16:Y=8'b1110\_1111;

7'd17:Y=8'b11101111;

7'd18:Y=8'b11011111;

7'd19:Y=8'b10111111;

7'd20:Y=8'b1011\_1111;

7'd21:Y=8'b11111111;

7'd22:Y=8'b1101\_1111;

7'd23:Y=8'b11111111;

7'd24:Y=8'b1011\_1111;

7'd25:Y=8'b1011\_1111;

7'd26:Y=8'b1011\_1111;

7'd27:Y=8'b1101\_1111;

7'd28:Y=8'b1110\_1111;

7'd29:Y=8'b1110\_1111;

7'd30:Y=8'b1110\_1111;

7'd31:Y=8'b1111\_1111;

7'd32:Y=8'b1110\_1111;

7'd33:Y=8'b1110\_1111;

7'd34:Y=8'b1110\_1111;

7'd35:Y=8'b1101\_1111;

7'd36:Y=8'b1101\_1111;

7'd37:Y=8'b1111\_1111;

7'd38:Y=8'b1011\_1111;

7'd39:Y=8'b1011\_1111;

7'd40:Y=8'b1011\_1111;

7'd41:Y=8'b1111\_1111;

7'd42:Y=8'b1101\_1111;

7'd43:Y=8'b1111\_1111;

7'd44:Y=8'b1110\_1111;

7'd45:Y=8'b1110\_1111;

7'd46:Y=8'b1110\_1111;

7'd47:Y=8'b1110\_1111;

7'd48:Y=8'b0111\_1111;

7'd49:Y=8'b0111\_1111;

7'd50:Y=8'b0111\_1011;

7'd51:Y=8'b1011\_1111;

7'd52:Y=8'b1011\_1111;

7'd53:Y=8'b1101\_1111;

7'd54:Y=8'b1101\_1111;

7'd55:Y=8'b1011\_1111;

7'd56:Y=8'b1110\_1111;

7'd57:Y=8'b1110\_1111;

7'd58:Y=8'b1110\_1111;

7'd59:Y=8'b1110\_0111;

default:Y=8'b1111\_1111;

endcase

end

end

endmodule

**数码管显示模块**

**Display.v**

module display(

input clk,

input scan\_sgn,

input start,

output reg [3:0] an,

output reg [7:0] seg,

input [7:0] Y,

input OE

);

integer i;

reg [1:0] a;

reg [7:0] display\_num [3:0] ;

always@(posedge clk or posedge start)

begin

if(start)

begin

a<=2'b00;

end

else if(scan\_sgn)

begin//这个begin真的要加。。不然真的会出错。。。

if(a==2'b11)

a<=2'b00;

else

a<=a+2'b1;

end

end

always@(posedge clk or posedge start)

begin

if(start)

seg<=8'b1111\_1111;

else

case(a)

2'b00:

begin

an<=4'b0111;

seg<=display\_num[3];

end

2'b01:

begin

an<=4'b1011;

seg<=display\_num[2];

end

2'b10:

begin

an<=4'b1101;

seg<=display\_num[1];

end

default:

begin

an<=4'b1110;

seg<=display\_num[0];

end

endcase

end

always@(posedge clk or posedge start)

if(start)

for(i=0;i<4;i=i+1)

display\_num[i]<=8'b1111\_1111;

else if(OE)

begin

display\_num[0]<=display\_num[1];

display\_num[1]<=display\_num[2];

display\_num[2]<=display\_num[3];

display\_num[3]<=Y;//最高位为最新一位

end

endmodule

**将按键转化为脉冲的模块：**

**Signal\_generate.v**

module signal\_generate(

input signal\_in,

input clk,

input reset,

output signal\_out\_pos,

output signal\_out\_neg

);

reg signal\_in1,signal\_in2,signal\_in3;

always @(posedge clk or posedge reset)

if(reset)

begin

signal\_in1 <= 0;

signal\_in2 <= 0;

signal\_in3 <= 0;

end

else

begin

signal\_in1 <= signal\_in;

signal\_in2 <= signal\_in1;

signal\_in3 <= signal\_in2;

end

assign signal\_out\_neg = signal\_in3 & ~signal\_in2;

assign signal\_out\_pos = signal\_in2 & ~signal\_in3;

endmodule

**top.v**

module top(

input clk,

output [7:0] seg,//注，这个必须写成wire类型！

output [3:0] an,

input [2:0] sw,

output [2:0] level,

output [4:0] rank,

output speaker,

input btnr,

input btnd,

input btnl,

input btnu,

input btns,

input stop

);

wire scan\_sgn;

wire OE;

wire [7:0] Y;

wire [9:0] timeunit;

music u\_music(

.stop (stop ),

.clk (clk ),

.start (btns ),

.speaker (speaker ),

.timeunit (timeunit )

);

scan u\_scan(

.start (btns ),

.clk (clk ),

.scan\_sgn (scan\_sgn )

);

difficultysel u\_difficultysel(

.start (btns ),

.clk (clk ),

.difficulty (sw ),

.timeunit (timeunit )

);

display u\_display(

.start (btns ),

.clk (clk ),

.scan\_sgn (scan\_sgn ),

.an (an ),

.seg (seg ),

.Y (Y ),

.OE (OE )

);

mom u\_mom(

.stop (stop ),

.clk (clk ),

.Y (Y ),

.start (btns ),

.OE (OE ),

.timeunit (timeunit )

);

judge u\_judge(

.stop (stop ),

.clk (clk ),

.btnu (btnu ),

.btnd (btnd ),

.btnr (btnr ),

.btnl (btnl ),

.start (btns ),

.Y (Y ),

.OE (OE ),

.level (level ),

.rank (rank ),

.timeunit (timeunit )

);

Endmodule

Ucf：

Net "seg<7>" LOC = T17;

Net "seg<6>" LOC = T18;

Net "seg<5>" LOC = U17 ;

Net "seg<4>" LOC = U18 ;

Net "seg<3>" LOC = M14 ;

Net "seg<2>" LOC = N14;

Net "seg<1>" LOC = L14;

Net "seg<0>" LOC = M13;

Net "an<0>" LOC = N16;

Net "an<1>" LOC = N15;

Net "an<2>" LOC = P18;

Net "an<3>" LOC = P17;

Net "rank<0>" LOC = U16;

Net "rank<1>" LOC = V16;

Net "rank<2>" LOC = U15;

Net "rank<3>" LOC = V15;

Net "rank<4>" LOC = M11;

Net "level<0>" LOC = N11;

Net "level<1>" LOC = R11;

Net "level<2>" LOC = T11;

Net "stop" LOC = M8;

Net "sw<0>" LOC = T10;

Net "sw<1>" LOC = T9;

Net "sw<2>" LOC = V9 ;

Net "clk" LOC=V10;

Net "btns" LOC = B8;

Net "btnu" LOC = A8;

Net "btnl" LOC = C4;

Net "btnd" LOC = C9;

Net "btnr" LOC = D9;

Net "speaker" LOC=T12;