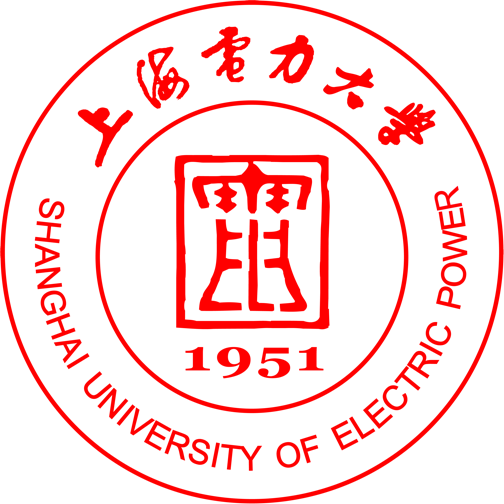
**上海电力大学**

**实践课程报告**



学 院：电子与信息工程学院

专 业：电子科学与技术

课程编号：2614110 课程名称：FPGA应用课程设计

学生姓名： 学号： 班级：

指导老师：赵倩、扬程

1919年8月10日

成绩：

|  |
| --- |
| 教师评语： |

**一、设计目的**

本课程的授课对象是电子科学与技术专业本科生，是电子类专业的一门重要的实践课程，是理论与实践相结合的重要环节。

本课程有助于培养学生使用Verilog硬件语言进行数字电路设计的能力，掌握模块划分、模块设计，模块仿真和整体仿真调试的方法，为以后从事硬件电路设计、仿真与调试工作打下坚实的基础。

**二、设计内容、要求及组织形式**

本课程要求学生在掌握数字电路理论的基础上，以FPGA器件为调试平台，以Verilog为硬件语言，以Quartus II为仿真下载工具，设计题目为洗衣机控制器，该实践课程以查阅文献资料、硬件设计、仿真调试、书写报告和独立答辩为基本组织形式。

**三、设计任务及要求**

用于模仿洗衣机的工作过程，完成洗衣机的核心控制功能。功能要求：洗衣机有正常洗、羊毛洗、快洗三种功能。其中，正常洗：待机2s，进水3s，正转10s，待机1s，反转10s，之后延续待机、正转、待机、反转的循环，循环7次。羊毛洗：待机2s，进水3s，正传5s，待机1s，反转5s，之后延续待机、正转、待机、反转的循环，循环10次。快洗：待机2s，进水3s，正传10s，待机1s，反转10s，之后延续待机、正转、待机、反转的循环，循环5次。数码管显示当前剩余秒数，LED灯或数码管显示当前状态。

正常洗除默认模式外，还能够设定进水时间（进水量），正、反转时间和循环次数。

洗衣机在任何模式下都能够暂停，以及从暂停状态下启动继续未完成的工作。还能够打断工作状态进入待机，并重新设定工作模式。

洗衣机完成当前工作后，能够报警告知用户。

拨码开关表示洗衣机是否盖上上盖以及是否有进水，若启动后洗衣机未盖上上盖或者无进水，则报警自动暂停，待盖上上盖或打开进水开关后，点击启动按钮，则洗衣机继续工作。

**四、设计原理**

洗衣机程序是一个用 Verilog 语言编写的硬件描述语言程序。它使用了有限状态机的设计方法来模拟洗衣机的工作过程。

在这个程序中，洗衣机有四种工作状态： IDLE 、 WATER\_IN 、 FORWARD 和 REVERSE 。这些状态分别表示洗衣机处于空闲、进水、正转和反转状态。程序使用一个名为 state 的寄存器来表示当前的工作状态。

洗衣机还有四种工作模式： NORMAL\_WASH 、 WOOL\_WASH 和 QUICK\_WASH 。这些模式分别表示普通洗涤、羊毛洗涤和快速洗涤。程序使用一个名为 mode 的输入信号来选择当前的工作模式。

程序中还定义了两个寄存器 timer 和 number ，分别用于倒计时和表示剩余循环次数。此外，程序还定义了一个名为 warning 的输出信号，用于在循环完成后告知用户。

程序使用一个 always 块来描述洗衣机的工作过程。在 always 块中，程序根据当前的工作状态和工作模式来更新寄存器的值，并根据输入信号（如 start 、 pause 和 reset ）来控制洗衣机的启动、暂停和复位。

**五、设计步骤**

5.1 设计思路

将工作模式分为准备阶段和工作阶段。工作模式 ，以normal\_wash为例: 首先进入洗衣准备阶段 idle 2个时钟周期， water\_in 3个时钟周期，forward 10个时钟周期，idle 1个时钟周期，reverse 10个时钟周期， 接下来进入工作阶段: idle 2个时钟周期、forward 3个时钟周期、idle 1个时钟周期、reverse 10个时钟周期的循环，将工作阶段循环7次，该模式结束

5.2 功能仿真

正常洗 mode0 （见图1）

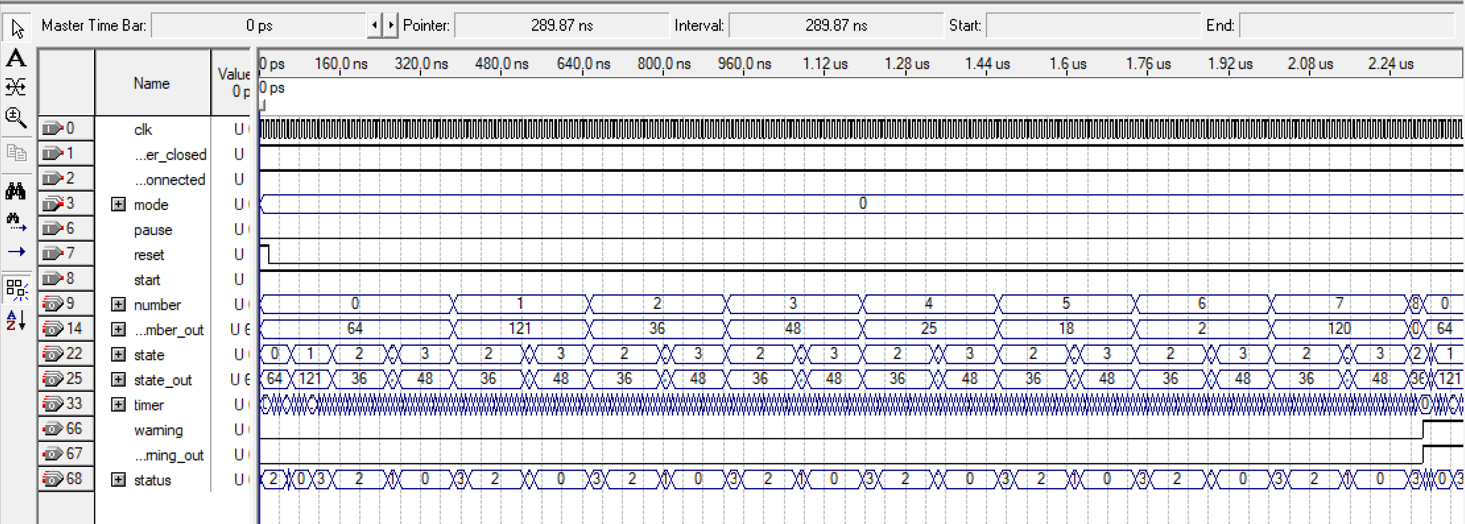


图1

羊毛洗 mode1 （见图2）

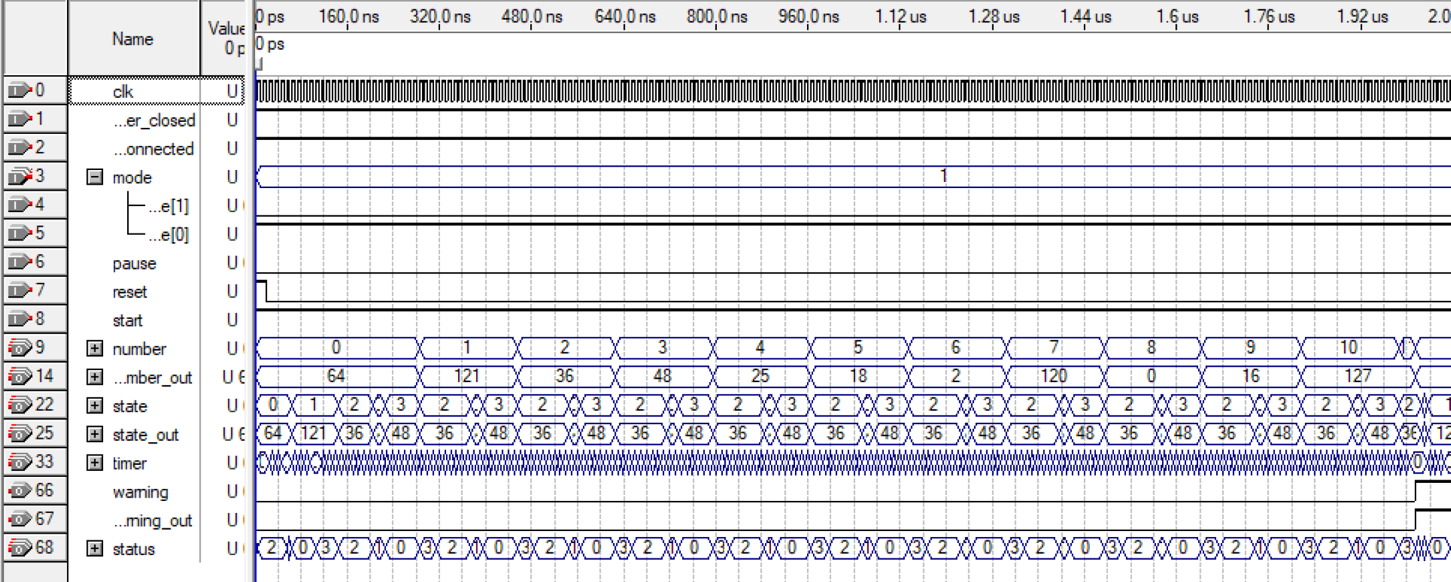


图2

快洗 mode2 （见图3）

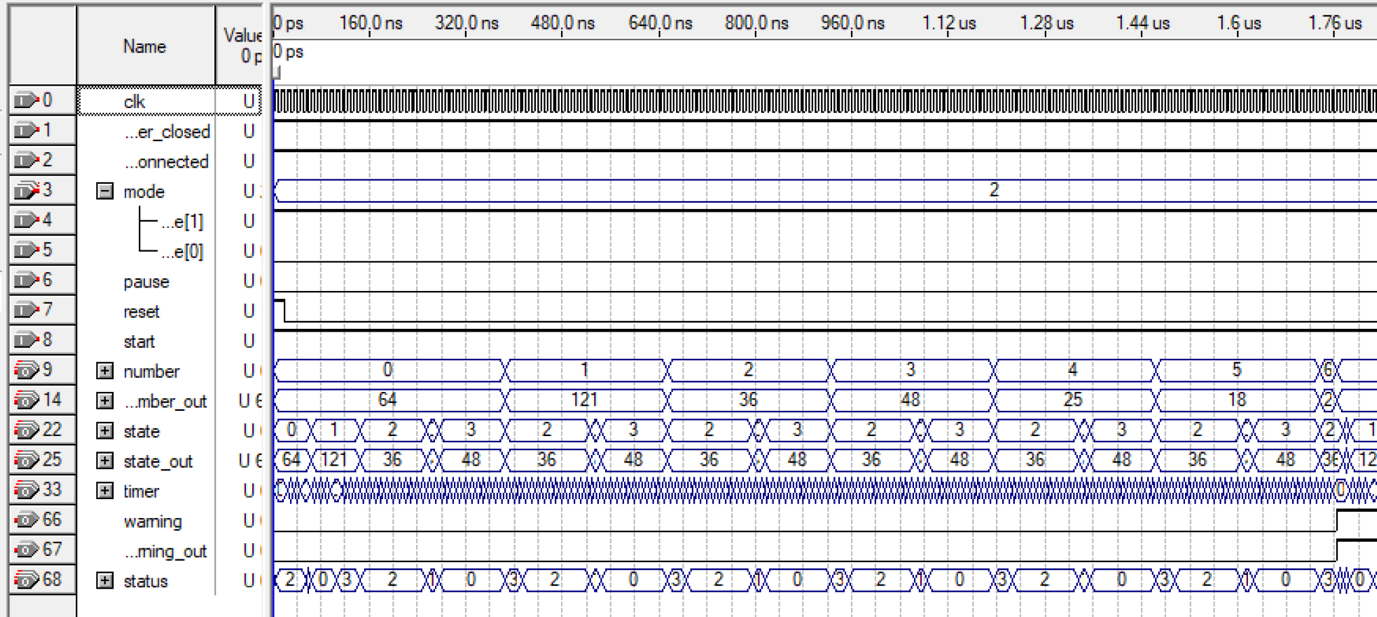


图3

若启动后洗衣机未盖上上盖(cover\_closed)或者无进水(water\_connected)或者按下暂停键(pause)，则报警(warning)，并自动暂停，待盖上上盖或打开进水开关或取消暂停后，点击启动按钮，则洗衣机继续之前的工作。（见图4）

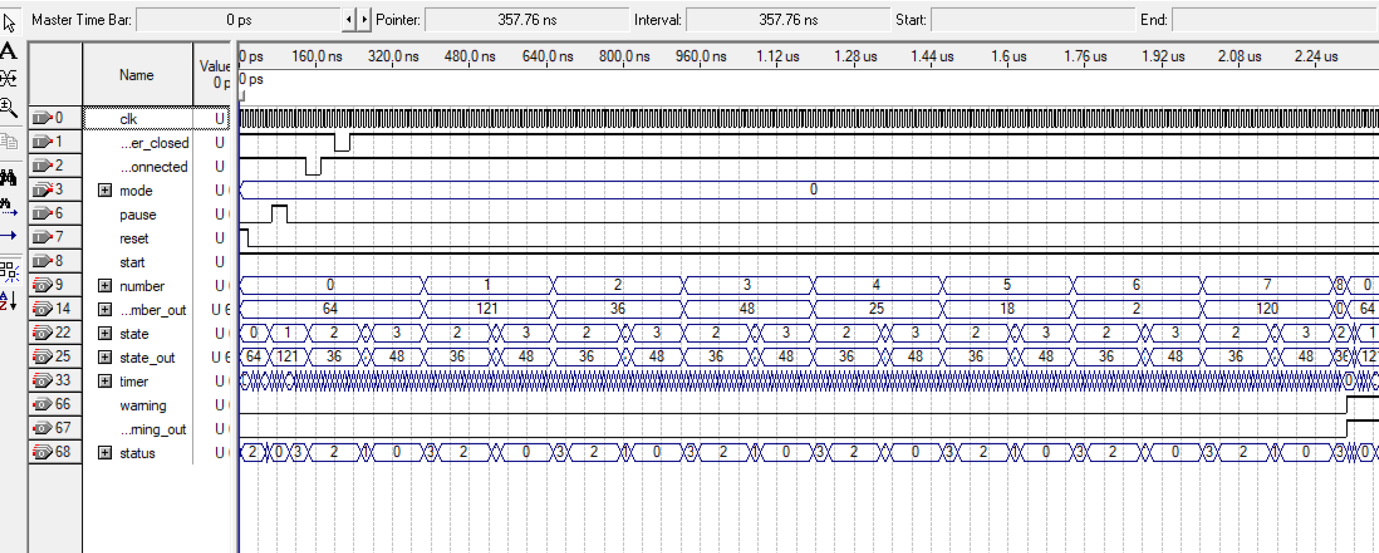


图4

5.3 verilog程序

用于模拟洗衣机的程序

module washing\_machine(

input clk,

input reset,

input cover\_closed,

input water\_connected,

input pause,

input start,

input [1:0] mode,

output reg [6:0] number\_out,

output reg [6:0] state\_out,

output reg [6:0] timer\_out,

output reg warning\_out

);

// 定义常量

localparam IDLE = 2'b00;

localparam WATER\_IN = 2'b01;

localparam FORWARD = 2'b10;

localparam REVERSE = 2'b11;

localparam NORMAL\_WASH = 2'b00;

localparam WOOL\_WASH = 2'b01;

localparam QUICK\_WASH = 2'b10;

// 定义变量

reg warning;

reg [31:0] counter;

reg [25:0] cnt;

reg preparing;

reg [2:0] status; // 定义 status 变量，用于记录工作状态的变化

reg [1:0] last\_state; // 定义 last\_state 变量，用于记录 state 变量的前一个值

reg start\_debouncing; // 启动消抖，避免启动时丢失status

reg clk\_out = 0;

reg [3:0] number;

reg [1:0] state;

reg [31:0] timer;

always @(posedge clk) begin //50MHz 分频 1 Hz

if (cnt < 25000000)

cnt <= cnt + 1;

else begin

clk\_out <= ~clk\_out;

cnt <= 0;

end

end

always @(posedge clk\_out) begin

if (reset) begin

state <= IDLE;

timer <= 0;

number <= 0;

counter <= 0;

warning <= 0;

preparing <= 1; // 初始状态为准备阶段

last\_state <= IDLE; // 初始状态为 IDLE

status <= 2; // 初始状态为 2

start\_debouncing <= 0;

end else if (start && cover\_closed && water\_connected) begin

warning <= 0; // turn off warning when starting

case (mode)

NORMAL\_WASH: begin

if (preparing) begin // 准备阶段

if (status == 2 && timer == 0 && start\_debouncing == 1) begin // 这里增加了对 status 的判断

state <= IDLE;

start\_debouncing <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && timer == 0) begin // 这里增加了对 status 的判断

state <= WATER\_IN;

timer <= 3;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && timer == 0) begin // 这里增加了对 status 的判断

preparing <= 0; // 进入工作阶段

start\_debouncing <= 0; // 重制启动消抖

status <= 3;//设置工作阶段循环次数。

end else if (timer > 0) begin

timer <= timer -1; // 计时器倒计时

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end else begin // 工作阶段

if (status == 3 && number < 8 && timer == 0 && start\_debouncing == 1) begin

state <= FORWARD;

timer <= 10;

status <= status - 1; // 更新 status 变量的值

end else if (status == 2 && number < 8 && timer == 0) begin

state <= IDLE;

timer <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && number <8 && timer == 0)begin

state<=REVERSE;

timer<=10;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && number<8 && timer == 0)begin

state<=FORWARD;

timer<=2;

number <= number + 1;//循环次数加一。

status<=3;//重置工作阶段状态。

end else if (number>=8 && timer == 0) begin

warning<=~warning;//模式结束，发出警告信号。

preparing<=1;//进入准备阶段。

number<=0;//重置循环次数。

status<=2;//重置准备阶段状态。

end else if(timer>0 && start\_debouncing == 1) begin

timer<=timer-1;//计时器倒计时。

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end // else: 工作阶段

end

WOOL\_WASH: begin

if (preparing) begin // 准备阶段

if (status == 2 && timer == 0 && start\_debouncing == 1) begin // 这里增加了对 status 的判断

state <= IDLE;

start\_debouncing <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && timer == 0) begin // 这里增加了对 status 的判断

state <= WATER\_IN;

timer <= 3;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && timer == 0) begin // 这里增加了对 status 的判断

preparing <= 0; // 进入工作阶段

start\_debouncing <= 0; // 重制启动消抖

status <= 3;//设置工作阶段循环次数。

end else if (timer > 0) begin

timer <= timer -1; // 计时器倒计时

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end else begin // 工作阶段

if (status == 3 && number < 11 && timer == 0 && start\_debouncing == 1) begin

state <= FORWARD;

timer <= 5;

status <= status - 1; // 更新 status 变量的值

end else if (status == 2 && number < 11 && timer == 0) begin

state <= IDLE;

timer <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && number < 11 && timer == 0)begin

state<=REVERSE;

timer<=5;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && number< 11 && timer == 0)begin

state<=FORWARD;

timer<=2;

number <= number + 1;//循环次数加一。

status<=3;//重置工作阶段状态。

end else if (number>= 11 && timer == 0) begin

warning<=~warning;//模式结束，发出警告信号。

preparing<=1;//进入准备阶段。

number<=0;//重置循环次数。

status<=2;//重置准备阶段状态。

end else if(timer>0 && start\_debouncing == 1) begin

timer<=timer-1;//计时器倒计时。

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end // else: 工作阶段

end

QUICK\_WASH: begin

if (preparing) begin // 准备阶段

if (status == 2 && timer == 0 && start\_debouncing == 1) begin // 这里增加了对 status 的判断

state <= IDLE;

start\_debouncing <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && timer == 0) begin // 这里增加了对 status 的判断

state <= WATER\_IN;

timer <= 3;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && timer == 0) begin // 这里增加了对 status 的判断

preparing <= 0; // 进入工作阶段

start\_debouncing <= 0; // 重制启动消抖

status <= 3;//设置工作阶段循环次数。

end else if (timer > 0) begin

timer <= timer -1; // 计时器倒计时

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end else begin // 工作阶段

if (status == 3 && number < 6 && timer == 0 && start\_debouncing == 1) begin

state <= FORWARD;

timer <= 10;

status <= status - 1; // 更新 status 变量的值

end else if (status == 2 && number < 6 && timer == 0) begin

state <= IDLE;

timer <= 1;

status <= status - 1; // 更新 status 变量的值

end else if (status == 1 && number <8 && timer == 0)begin

state<=REVERSE;

timer<=10;

status <= status - 1; // 更新 status 变量的值

end else if (status == 0 && number<6 && timer == 0)begin

state<=FORWARD;

timer<=2;

number <= number + 1;//循环次数加一。

status<=3;//重置工作阶段状态。

end else if (number>=6 && timer == 0) begin

warning<=~warning;//模式结束，发出警告信号。

preparing<=1;//进入准备阶段。

number<=0;//重置循环次数。

status<=2;//重置准备阶段状态。

end else if(timer>0 && start\_debouncing == 1) begin

timer<=timer-1;//计时器倒计时。

end else if (timer == 0 && start\_debouncing == 0) begin

timer <= 2; // 在启动区为第一个state的timer赋值

start\_debouncing <= 1;

end

end // else: 工作阶段

end

default: begin

// do nothing.

end

endcase // case(mode)

end else if (!cover\_closed || !water\_connected || pause) begin

warning <= ~warning;

state <= IDLE;

end

end // always @(posedge clk)

// 以下为输出显示模块

always @(\*) begin

case (state)

IDLE: state\_out = 7'b1000000; // display "0" on state\_out when in idle state

WATER\_IN: state\_out = 7'b1111001; // display "1" on state\_out when in water\_in state

FORWARD: state\_out = 7'b0100100; // display "2" on state\_out when in forward state

REVERSE: state\_out = 7'b0110000; // display "3" on state\_out when in reverse state

default: state\_out = 7'b1111111;

endcase

case (number)

4'b0000:number\_out=7'b1000000;

4'b0001:number\_out=7'b1111001;

4'b0010:number\_out=7'b0100100;

4'b0011:number\_out=7'b0110000;

4'b0100:number\_out=7'b0011001;

4'b0101:number\_out=7'b0010010;

4'b0110:number\_out=7'b0000010;

4'b0111:number\_out=7'b1111000;

4'b1000:number\_out=7'b0000000;

4'b1001:number\_out=7'b0010000;

4'b1010:number\_out=7'b0001000;

4'b1011:number\_out=7'b0000011;

default:number\_out=7'b0000000;

endcase

case (timer)

4'b0000: timer\_out=7'b1000000;

4'b0001: timer\_out=7'b1111001;

4'b0010: timer\_out=7'b0100100;

4'b0011: timer\_out=7'b0110000;

4'b0100: timer\_out=7'b0011001;

4'b0101: timer\_out=7'b0010010;

4'b0110: timer\_out=7'b0000010;

4'b0111: timer\_out=7'b1111000;

4'b1000: timer\_out=7'b0000000;

4'b1001: timer\_out=7'b0010000;

4'b1010: timer\_out=7'b0001000;

4'b1011: timer\_out=7'b0000011;

default: timer\_out=7'b0000000;

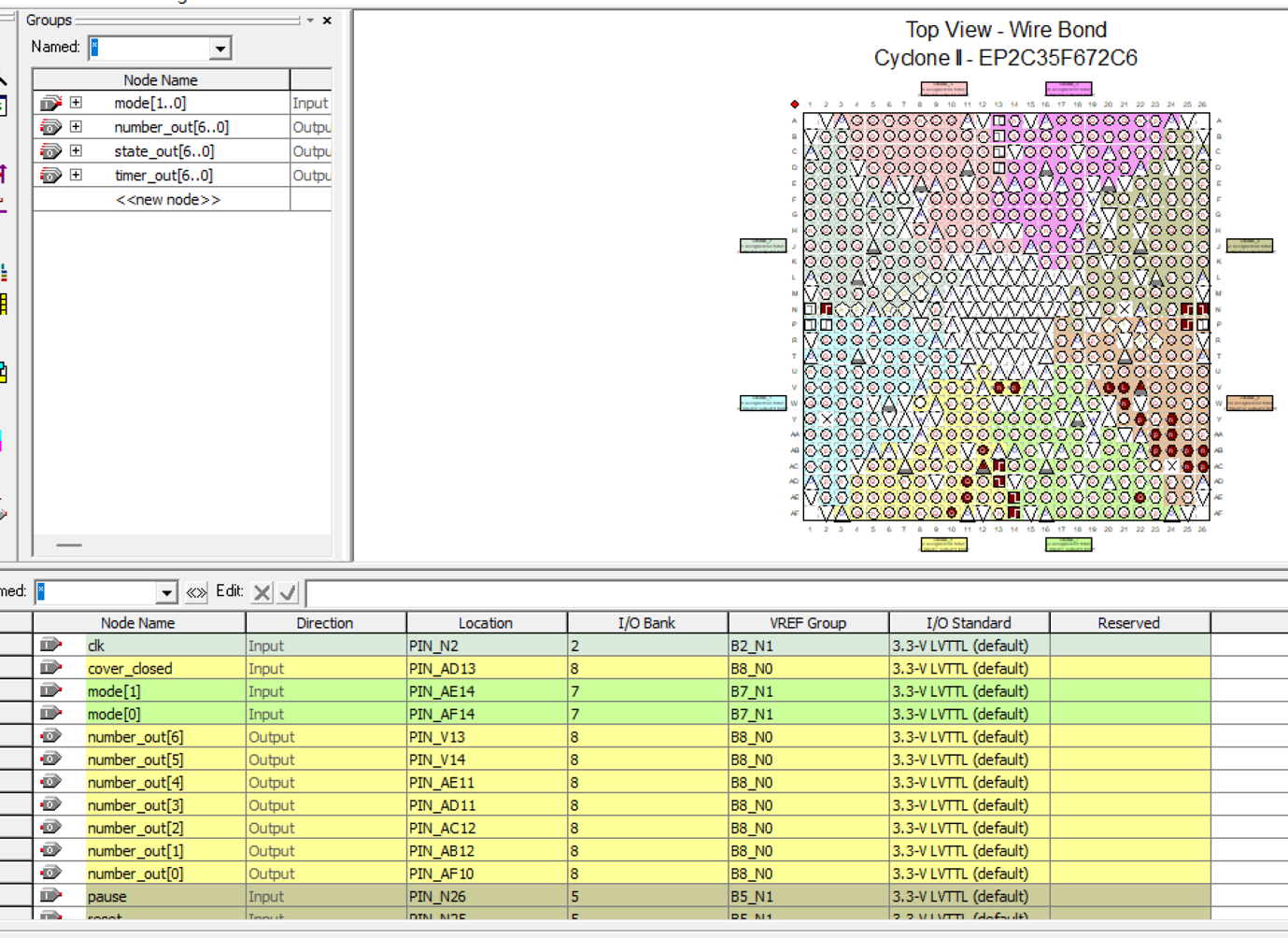
endcase

warning\_out = warning; // display warning on warning\_out

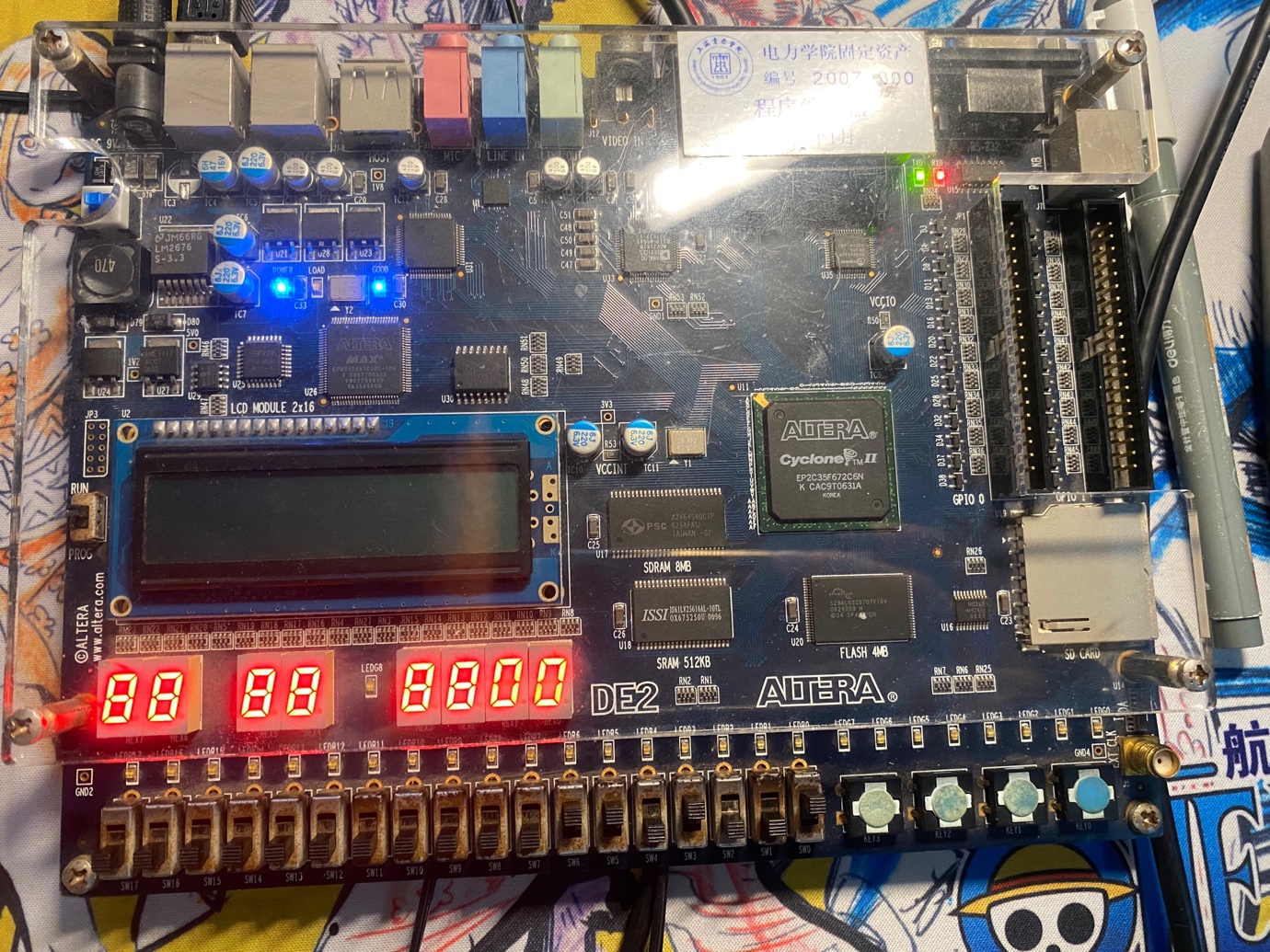
end

endmodule // washing\_machine

5.4 引脚功能分配



5.5 上机验证



**六、问题及解决**

1. 多驱动问题：在最初的洗衣机程序中，我们使用了两个 always 块来驱动同一个寄存器 `warning`。这会导致多驱动问题，从而导致仿真错误。我们通过将第二个 always 块中的代码移动到第一个 always 块中来解决这个问题。

2.仿真时间不够长：在运行仿真时，如果仿真时间不够长，那么可能无法看到完整的波形。我们可以通过增加仿真时间来解决这个问题。

3. 测试台模块设置不正确：在 Quartus II 中运行仿真时，如果测试台模块的设置不正确，那么可能无法正确地运行仿真。我们可以通过检查 "Edit Test Bench Setting" 对话框中的设置来解决这个问题。

**七、收获和体会**

在本次FPGA课程设计中，我完成了模拟洗衣机程序的verilog设计工作。这项任务让我深刻认识到数字电路设计的重要性，并让我体会到数字电路设计的原理、流程和技能。通过参与这个课程设计，我取得了以下几个收获和体会：

首先，我通过本次课程设计深入了解了FPGA的架构和设计方法，了解了FPGA开发的基本流程。 HDL语言设计和实现一个功能完整的数字电路模块。

其次，通过模拟洗衣机程序的Verilog设计，我深入了解了数字电路的基础知识和设计原则，程序编写过程中的各种问题，也锻炼了我的搜索能力。

最后，这项课程设计让我更加自信地使用数字电路设计工具，编写更加复杂的程序，加深了我对数字电路设计和FPGA开发的了解程度。

总体来说，本课程设计帮助我更好地认识和掌握数字电路设计的基本原理和实践技能，感谢老师的指导和支持！