**数字电路与逻辑设计**

**课程设计报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **团队成员姓名** | **班级** | **学号** | **贡献百分比** |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **课程设计实验部分：实验完成情况、时间**  **（亮点、完成、部分完成）** | | | **总分**  **（实验部分70% +报告30%）** |
|  | **设计1** | **设计2** |  |
| **检查结果** |  |  |
| **检查名次** |  |  |
| **检查老师** |  |  |

**报告人 ：**

**实验指导教师：**

**报告批阅教师：**

**计算机科学与技术学院**

**20 年 月 日**

**数字逻辑课程设计学生工作表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **班 级** | **姓 名** | **学 号** | | **验收时间（教师填写）** |
| ACM1501 | 吴肇敏 | U201514721 | |  |
|  |  |  | |
| **（学生填写）** | | | **课设进度记录（学生填写）** | |
| **各自承担课程设计具体工作量记录**  **1、姓名：吴肇敏**  （1）设计1：电梯  （2）设计2：xxx  **2、姓名：**  （1）设计1：xxx  （2）设计2：xxx  **3、难点、亮点** | | | 日期 | 进度 |
| 9.12 | 接受任务；  决定设计1的题目和二人的任务分配；  大体完成设计1的整体结构设计；  。。。 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **实验平台故障记录（学生填写，请注明实验平台的编号）** | |

**重要说明**

1. **时间安排：课内**2周。
2. **验收准备：**
   1. 完成本表学生应该填写部分；
   2. 同组的每位学生必须都能**以独自完成的方式**应对任何形式的验收；
   3. 完成课程设计报告书（**格式参见模板）**；
   4. 将源程序和报告的电子文档交班长。
3. **检查过程：** 
   1. 提交验收准备材料，请求老师验收，之后按验收老师的要求做；
   2. 在开发平台上根据验收老师的要求进行演示；
   3. 检查过程中独立回答老师提出的相关问题；
   4. 验收老师有权根据具体情况调整验收的内容与方式；
   5. 验收完成后关闭电源，整理好设备。
4. **评分标准：**
5. 同组者工作量的分配；
6. 在完成控制器基本要求外，有**亮点**为加分项；
7. 在规定时间内完成控制器基本要求；
8. 在规定时间内完成控制器**部分**基本要求；
9. 检查时间。
10. **课程设计判定为不合格的一些情形： （本人已阅读此条款1-5项：签名 ）**
    1. **请人代做或冒名顶替者；**
    2. **替人做且不听劝告者；**
    3. **课程设计报告内容抄袭或雷同者；**
    4. **课程设计报告内容与实际实验内容不一致者；**
    5. **课程设计代码抄袭者。**

目 录

[1 课程设计概述 7](#_Toc485257853)

[1.1 课设目的 7](#_Toc485257854)

[1.2 课设要求 7](#_Toc485257855)

[1.3 课设任务 7](#_Toc485257856)

[1.4 实验环境 7](#_Toc485257857)

[2 洗衣机控制系统设计 9](#_Toc485257858)

[2.1 目的 9](#_Toc485257859)

[2.2 内容 9](#_Toc485257860)

[2.3 设计思路 13](#_Toc485257873)

[2.4 代码实现 18](#_Toc485257876)

[2.5 仿真过程 26](#_Toc485257885)

[2.6 主要故障 33](#_Toc485257888)

[2.7 功能测试 35](#_Toc485257892)

[2.8 实验中遇到的主要问题及解决方法 39](#_Toc485257895)

[2.9 实验方案的改进意见 40](#_Toc485257896)

[3 总结与心得 46](#_Toc485257900)

[3.1 课设总结 46](#_Toc485257901)

[3.2 课设心得 46](#_Toc485257902)

[4 参考文献 47](#_Toc485257903)

[附 录1（源程序） 48](#_Toc485257904)

# 课程设计概述

## 课设目的

（1）掌握Vivado软件的使用方法；

（2）熟悉FPGA器件的使用方法；

（3）用Verilog HDL进行较复杂逻辑电路的设计和调试；

（4）学习数字系统的设计方法；

（5）通过规范化的实验报告，培养学生良好的文档习惯以及撰写规范文档的能力。

## 课设要求

（1）能够全面地应用课程中所学的基本理论和基本方法，完成从设计逻辑电路到设计简单数字系统的过渡；

（2）能力独立思考、独立查阅资料，独立设计规定的系统；

（3）能够独立地完成实施过程，包括电路设计、调试、排除故障、仿真和下载验证。

## 课设任务

（1）各组要制定出详细设计方案，明确成员各自分工，认真记载毕业设计工作日记；

（2）通过Verilog HDL完成规定的设计任务，采取模块化、层次化的设计方法设计电路，然后进行编译和仿真，认真记录实施过程中遇到的各自故障以及解决方法，保证设计的正确性；

（3）生成bit文件，下载到开发板上，通过实际线路进行验证设计的正确性；

（4）撰写设计报告，并对存在的问题进行分析、提出改进意见。

## 实验环境

开发环境为Vivado 2015.2软件和开发板NEXYS 4（芯片为XC7A100TCSG324-1，封装为CSG3242）。Vivado 2015.2是使用Xilinx FPGA必备的设计工具。它可以完成FPGA开发的全部流程，包括设计输入、仿真、 综合、布局布线、生成bit文件、配置以及在线调试等功能。

Nexys4 开发板简介：参见图1-1所示，它是一款简单易用的数字电路开发平台，可以支持在课堂环境中来设计一些行业应用。大规模、高容量的FPGA，海量的外部存储，各种USB、以太网、以及其它接口、这些让Nexys4-DDR能够满足从入门级组合逻辑电路到强大的嵌入式系统的设计。同时，板上集成的加速度、温度传感器，MEMs数字麦克风，扬声器放大器以及人量的I/O设备，让Vexys4-DDR不需要增添额外组件而用于各种各样的设计。

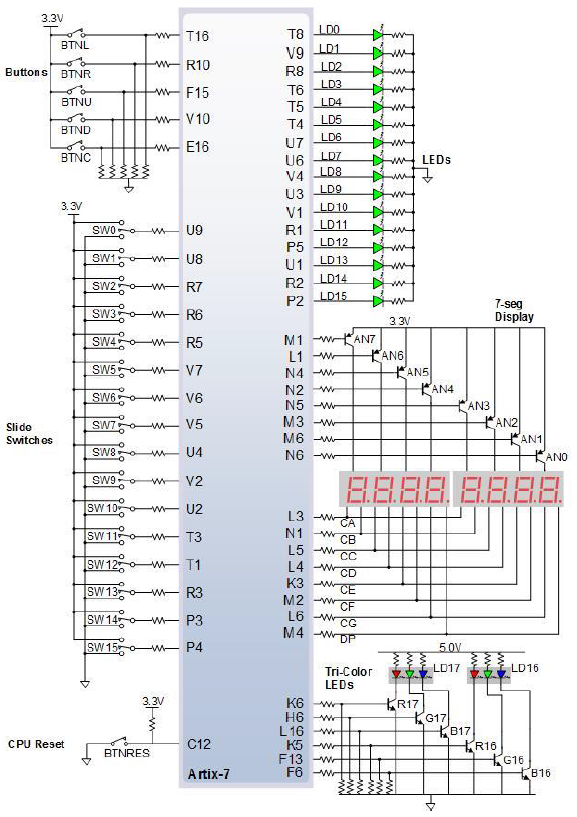


图 1-1 Nexys4通用I / O设备

# 洗衣机控制系统设计

## 目的

（1）掌握较复杂的逻辑设计和调试

（2）掌握用原理图+Verilog HDL语言设计逻辑电路；

（3）学习数字系统设计方法；

（4）掌握Vivado软件的使用方法；

（5）熟悉FPGA器件的使用。

## 内容

### 电源开关

电源开关同时作为电路总清零信号（Reset），当Reset = Off不工作、为On时电路进入初始状态，且电源指示灯亮（又称洗衣机控制器运行指示灯）,控制器初始状态为：“洗漂脱”。

### 启动/暂停

启动：在电源打开，“启动”指示灯不亮的情况下，按一下“启动/暂停”，“启动”指示灯亮，洗衣机开始工作或恢复暂停时的工作状态并开始工作；

暂停：在电源打开，“启动”指示灯亮的情况下，按一下“启动/暂停”，“启动”指示灯灭，此时洗衣机处于“暂停”状态，等待重新“启动”。

暂停后可以重新选择洗衣程序。

### 洗衣机洗衣程序

（1）时间设置

进水为3分钟（为了方便描述假定衣物重量为3kg）；洗衣9分钟； 排水3分钟；脱水/甩干3分钟，漂洗6分钟。

说明：进水时间随衣物重量改变，这里假定衣物重量分为5kg、4kg、3kg、2kg 4种情况，控制器的“水位”控制分别对应为进水5、4、3、2分钟。

（2）洗涤程序

进水---洗衣，共12分钟；正在进行洗涤时，洗涤灯闪烁，正在进水时，进水灯亮。

（3）漂洗程序

排水---甩干---进水---漂洗，共15分钟，正在进行漂洗时，漂洗灯闪烁，正在进水时，进水灯亮，正在进行排水和脱水/甩干时，对应指示灯亮。

（4）脱水程序

排水---甩干，共6分钟。正在进行排水时，排水灯闪烁，正在进行脱水/甩干时，脱水灯亮。

### 洗衣模式

有以下6种洗衣模式可供选择。

为了方便提示洗衣过程，假定未完成的洗衣过程其对应的指示灯亮，正在进行的洗衣过程其对应的指示灯闪烁，已完成的洗衣过程前对应指示灯熄灭。

（1）洗漂脱

它含有“洗涤”、“漂洗”、“脱水”3个过程，整个程序流程包括：洗涤12分钟（进水---洗衣）---漂洗15分钟（排水---甩干---进水---漂洗）----脱水6分钟（排水---甩干），共33分钟。

（2）单洗（仅洗涤）

它含有洗涤1个过程，整个程序流程包括：洗涤12分钟（进水---洗衣）。

（3）洗漂

它含有“洗涤”、“漂洗”2个过程。整个程序流程包括：洗涤12分钟（进水---洗衣）---漂洗15分钟（排水---甩干---进水---漂衣），共27分钟；

（4）单漂（仅漂洗）

它含有“漂洗”1个过程。整个程序流程包含：漂洗15分钟（排水---甩干---进水---漂衣）。

（5）漂脱

它含有“漂洗”、“脱水”2个过程。整个程序流程包含：漂洗15分钟（排水---甩干---进水---漂衣）----脱水6分钟（排水---甩干），共21分钟。

（6）单脱（仅脱水）

它含有“脱水”1个过程。整个程序流程包含：脱水6分钟（排水---甩干）。

为了便于演示，实际设计上述时间均以秒为单位。

### 模式选择

系统共有6种洗衣模式供选择，初始状态为“洗漂脱”洗衣模式，每按一次模式选择，洗衣模式改变一次，改变顺序为：“洗漂脱”---“单洗”----“洗漂”-----“单漂”-----“漂脱”-----“单脱”----“洗漂脱”循环。

在选择洗衣模式时，总剩余时间7段数码管会显示新的洗衣模式的总时间，当前模式时间7段数码显示管会显示当前所选洗衣模式中第一个洗衣程序的所需时间。同时“洗涤”、“漂洗”和“脱水”指示灯也随着新的洗衣模式的变化而变化。某个指示灯亮，表示在本洗衣模式中包含对应的工作流程；反之则不包含。

洗衣模式选择完毕，按启动按钮，该洗衣模式启动，当该洗衣模式结束时要发出报警声，并回到洗漂脱模式。

### 水位

控制器会根据衣物重量（假定衣物重量分为5kg、4kg、3kg、2kg 4种情况）自动选择“水位”，水位由2个7段数码显示管指示。

### 蜂鸣

每按一次按钮时，蜂鸣指示灯闪烁一次，在每个程序结束时，蜂鸣指示灯闪烁5次，同时“启动”指示灯熄灭。

### 关闭

每次所选的洗衣模式结束全部工作后，5秒内无操作，自动执行“电源”开关关闭操作。

### 儿童锁（亮点）

在洗衣过程中，很可能由于儿童的误操作导致洗衣过程终止，造成了不必要的麻烦。在这样的背景下，设计了儿童锁功能。打开儿童锁开关后，洗衣机将保持原先的工作状态，不受电源等开关的控制，直到洗衣过程结束或关闭儿童锁按钮。

### 防误触（亮点）

在实际操作过程中，有时会由于误按按钮导致机器断电，只能重新开始洗衣过程，造成了不必要的麻烦。在这样的背景下，设计了防误触功能。在关闭电源开关之后在1秒内打开电路仍可以照常继续工作，但若超过1秒则设备关闭。

### 预约（亮点）

在实际使用过程中，常常由于无法准时操作，需要预约一定时间后自动开启洗衣。于是设计了预约功能，通过若干个开关控制预约时间后，洗衣机将在预约时间后开始运行。

### 输入输出

输入为系统时钟、电源开关、暂停开关、洗衣模式选择按钮、儿童锁开关、衣物重量设置开关；输出为电源指示灯、运行指示灯、蜂鸣指示灯、洗涤灯、漂洗灯、脱水灯、进水灯、排水灯、洗涤模式灯、漂洗模式灯、脱水模式灯。其中，洗涤模式包括进水、洗衣两个过程，漂洗模式包括排水、脱水、进水、漂洗四个过程，脱水模式包括排水、脱水两个过程。对于模式的具体情况详见“2.2.4洗衣模式”。洗衣机控制的输入输出见如图2-1所示。

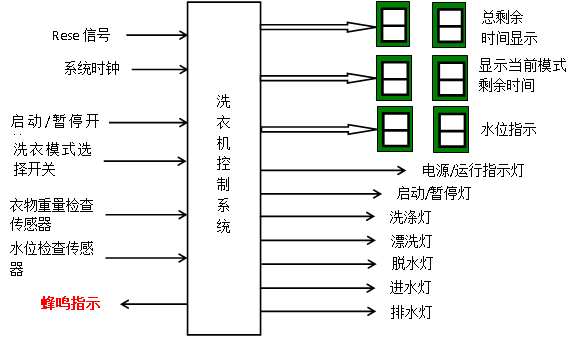


图2-1 洗衣机控制系统结构框图

## 设计思路

### 模块结构

整个系统的层次结构如图2-2所示。顶层模块为main，完成主要的控制功能，调用了divider、countdown、sel\_mode三个模块，其中divider完成分频功能，countdown实现倒计时功能，sel\_mode实现模式选择功能。countdown进一步调用format\_disp模块，format\_disp模块实现了3个2位数字格式显示器。format\_disp模块进一步调用disp\_num模块，disp\_num模块实现了显示8个数码管对应的数字。disp\_num调用了num2seg和show\_seg，其中num2seg实现了数字对7位数码管信号的转化，show\_seg实现了对数码管信号的显示。

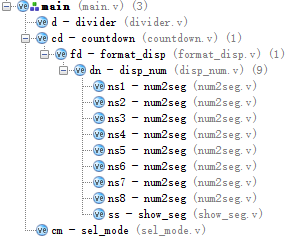


图2-2 系统模块层次图

### 各模块实现思路

1. main

主模块的主体是2类有限状态机，第一类状态机为mode\_state，只有一个，共有10个状态，其中0状态代表预暂停，1-6状态代表6个洗衣模式，7状态代表关机，8状态代表暂停，9状态代表预约，状态图如图2-3所示。第二个状态机为work\_state，共有6个，分别表示6个洗衣模式下的运行状态，在每个模式下状态的数量、顺序均不尽相同。以模式1洗漂脱为例，共有10个状态，其中1-8状态代表了1-8个工作步骤，0状态代表预置状态，其它代表结束状态，状态图如图2-4所示。

主模块首先检测预约按钮是否按下，以及预约时间，调用分频器模块（divider）分频实现统一的1秒的时钟。之后调用倒计时模块（countdown）进行预约倒计时。当收到倒计时模块的时间到信号时，跳转到暂停状态。在此调用sel\_mode，使用户通过按钮来控制将要执行的洗衣状态，并通过开关控制将要洗的衣服重量。重置三个时间，向倒计时模块（countdown）发送开始计时信号，开始洗衣。每收到一次time\_up信号，即跳转到下一个洗衣步骤（work\_state状态），使步骤灯跳转和蜂鸣灯闪烁一次。最初所有将要执行的状态灯均亮，每次第一个亮着的状态灯闪烁，每执行完一个模块，该模块的状态灯熄灭。直到最洗衣过程结束，进行倒计时，使蜂鸣灯闪烁。当倒计时结果结束后，自动强制将状态变为断电状态。



图2-3 主模块mode\_state状态机变化

主模块下每个一级状态（mode\_state）对应若干个二级状态（work\_state），每个二级状态代表一个洗衣小步骤。每个小步骤是洗衣过程的最小单元，若干个小步骤可构成洗衣模式。整体设计思路为Moore型电路，以分频后的1Hz时钟为周期，对每个小步骤状态进行处理，可以很好的完成洗衣机的功能。



图2-4 主模块work\_state状态机变化

1. divider

输入为系统时钟（clk），输出为分频时钟（cp）。利用计数器对系统100MHz的时钟进行统计，当统计到50 000 000时分频时钟cp翻转，即可得到1Hz频率的时钟。

1. sel\_mode

输入为模式选择按钮（btn），输出为切换到的模式（next\_mode）。初始的next\_mode为1，之后每按一下按钮，即收到一个btn的上升沿信号，若next\_mode < 6，则next\_mode加1，否则next\_mode变为1，实现循环计数。

1. countdown

输入为系统时钟（clk），分频后时钟（cp），重置计时（reset），暂停计时（hold\_wire），三个初始的数字（num1~num3），关闭计时器（clear）；输出为计时完成信号（time\_up），数码管显示内容（show\_num），数码管显示信息（show\_port）。

在每次时钟cp上升沿进行判断，若收到一个reset信号，则将对应的位置置为num1~num3中对应的数字；若收到hold\_wire信号，则暂停计时，收到cp信号不作处理；若收到clear信号，则将clear信号传递给下一级模块，清除数码管内容；若没有收到上述信息，则将num1和num2同时减1，由于衣物重量只与重量检测器有关，因此num3不变。

1. format\_disp

输入为系统时钟（clk），三个显示的数字（num1~num3），关闭计时器信号（clear）；输出为数码管显示内容（show\_num），数码管显示信息（show\_port），将显示信息传到上一级。

该模块将第3、6个数码管置为一根短横线，第1-2、4-5、7-8个数码管分别显示3个输入的2位数。计算出每一位对应的数字后，将信息传导下一级模块disp\_num，将显示信息传到上一级。

1. disp\_num

输入为系统时钟（clk），8个显示的数字（num1~num8），关闭计时器信号（clear）；输出为数码管显示内容（show\_num），数码管显示信息（show\_port）。

该模块调用num2seg模块，将8个数字全部转化为7段数码管所能识别的信息之后，调用show\_seg进行显示，将显示信息传到上一级。

1. num2seg

输入为一个十进制数字num，输出为一个数码管信号seg。基本实现思路为一个多路选择器。

1. show\_seg

输入为系统时钟（clk），8个显示的数字的数码管信号（num1~num8），关闭计时器信号（clear）；输出为数码管显示内容（show\_num），数码管显示信息（show\_port）。

利用高频时钟对8个数码管快速扫描，实现每个数码管显示不同的数字。

## 代码实现

洗衣机顶层原理图，参见图2-5所示。

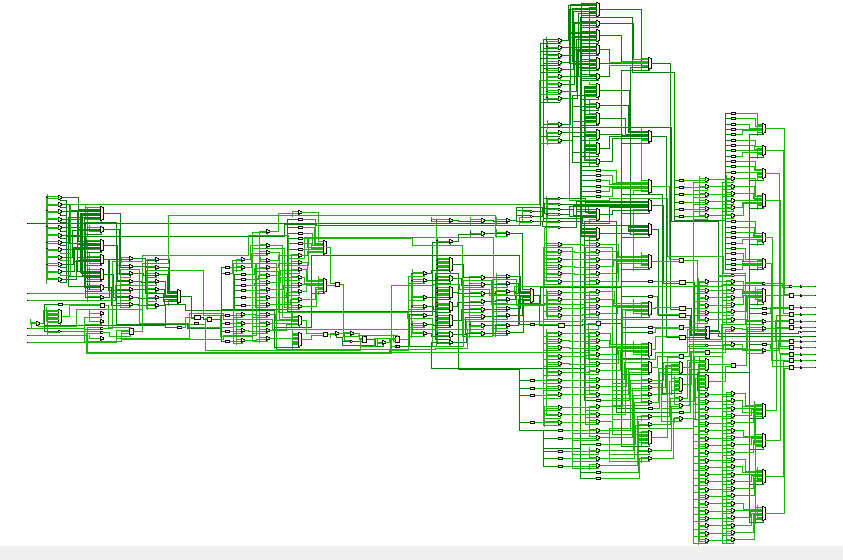


图2-5 洗衣机顶层原理图

### 顶层模块main（总状态控制）Verilog代码

|  |
| --- |
| 程序2-1 顶层模块main（总状态控制）Verilog代码 |
| module main(  //inputs  input reset,  input clk,  input pause,  input choose\_mode, // button  input [1:0] clo\_weight,  input child,  //outputs  output reg bee,  output [6:0] show\_num,  output [7:0] show\_port,  output reg power\_on,  output reg start,  output reg washing,  output reg rinsing,  output reg drying,  output reg water\_in,  output reg water\_out,  output reg final\_mode\_wash,  output reg final\_mode\_rinse,  output reg final\_mode\_dry );  divider d (clk, cp);  countdown cd (clk, cp, recount, hold, num1, num2, num3, time\_up, show\_num, show\_port, clear);  sel\_mode cm (choose\_mode, next\_mode);  always @(\*) begin  case (mode\_state)  0: begin //pre pausing mode  // initial varibles …  1: begin //washing + rinsing + drying  case (work\_state):  0: begin //start  end  1: begin //water in  end  2: begin //washing  end  3: begin //water out  end  4: begin //drying  end  5: begin //water in  end  6: begin //rinsing  end  7: begin //water out  end  8: begin //drying  end  default: begin //ending  end  endcase  2: begin //washing  case (work\_state):  0: begin //start  end  1: begin //water in  end  2: begin //washing  end  default: begin  end  endcase  3: begin //washing + rinsing  case (work\_state):  0: begin //start  end  1: begin //water in  end  2: begin //washing  end  3: begin //water out  end  4: begin //drying  end  5: begin //water in  end  6: begin //rinsing  end  default: begin  end  endcase  4: begin //rinsing  case (work\_state):  0: begin //start  end  3: begin //water out  end  4: begin //drying  end  5: begin //water in  end  6: begin //rinsing  end  default: begin //ending  end  endcase  end  end  5: begin //rinsing + drying  case (work\_state)  0: begin //start  end  3: begin //water out  end  4: begin //drying  end  5: begin //water in  end  6: begin //rinsing  end  7: begin //water out  end  8: begin //drying  end  default: begin  end  endcase  end  end  6: begin //drying  case (work\_state)  0: begin //start  end  7: begin //water out  end  8: begin //drying  end  default: begin  end  endcase  end  end  7: begin //power off  end  8: begin //pausing mode  end  9: begin //reserve mode  end  endcase  end  endmodule |

### 倒计时模块countdown（用于产生倒计时）Verilog代码

|  |
| --- |
| 程序2-2 倒计时模块countdown（用于产生倒计时）Verilog代码 |
| module countdown(  input clk\_k, // original 100MHz  input clk, // 1Hz clock  input [1:0] reset, // reset[0] : first number , reset[1] : second number  input hold\_wire, // 1~3 : 3 numbers hold  input [7:0] num1,  input [7:0] num2,  input [7:0] num3, // constant  output reg [1:0] final\_time\_up,  output [6:0] show\_num,  output [7:0] show\_port,  input clear  );  always @(\*) final\_time\_up = time\_up & pause;  format\_disp fd (clk\_k, w1, w2, w3, show\_num, show\_port, clear);  assign w1 = next1;  assign w2 = next2;  assign w3 = num3;    // if not reset, count the number  always @(posedge clk) begin  // if not reset, count the number  // else reset the number  end  // calculate final\_time\_up  always @(negedge clk) begin  if ((!hold\_wire) && reset == 'b00 && (next1 == 0 || next2 == 0))  pause <= 'b11;  else pause <= 'b00;  end  // get time\_up when countdown finished  always @(posedge clk) begin  // calculate time\_up  end  endmodule |

### 选择模式模块sel\_mode（选择洗衣模式）Verilog代码

|  |
| --- |
| 程序2-3 选择模式模块sel\_mode（选择洗衣模式）Verilog代码 |
| module sel\_mode(  input btn,  output reg [3:0] next\_mode  );  initial begin  next\_mode = 1;  end  always @(posedge btn) begin  if (next\_mode == 6) begin  next\_mode <= 1;  end  else begin  next\_mode <= next\_mode + 1;  end  end  endmodule |

### 分频器模块divider（系统时钟分频）Verilog代码

|  |
| --- |
| 程序2-4 分频器模块divider（系统时钟分频）Verilog代码 |
| module divider(clk, clk\_N);  input clk;  output reg clk\_N;  parameter dely500us = 50\_000\_000;  //debug  //parameter dely500us = 8;  reg [31:0] counter;  initial begin  clk\_N = 0;  counter = 0;  end  always @(posedge clk) begin  if(counter==dely500us) begin  clk\_N <= ~clk\_N; // reverse the clock cp  counter <= 0;  end  else begin  counter <= counter + 1;  end  end  endmodule |

### 格式化输出模块format\_disp（将三个数字格式化为8个数字信号）Verilog代码

|  |
| --- |
| 程序2-5 格式化输出模块format\_disp（将三个数字格式化为8个数字信号）Verilog代码 |
| module format\_disp(  input clk, // original clock 100MHz  input [7:0] num1,  input [7:0] num2,  input [7:0] num3,  output [6:0] show\_num,  output [7:0] show\_port,  input clear  );  wire [3:0] n1, n2, n3, n4, n5, n6, n7, n8;  assign n1 = num1 / 10;  assign n2 = num1 % 10;  assign n3 = 11; //invalid  assign n4 = num2 / 10;  assign n5 = num2 % 10;  assign n6 = 11; //invalid  assign n7 = num3 / 10;  assign n8 = num3 % 10;  disp\_num dn (clk, n1, n2, n3, n4, n5, n6, n7, n8, show\_num, show\_port, clear);  endmodule |

### 显示数字模块disp\_num（将8个数字信号分别显示）Verilog代码

|  |
| --- |
| 程序2-6 显示数字模块disp\_num（将8个数字信号分别显示）Verilog代码 |
| module disp\_num(  input clk,  input [3:0] num1,  input [3:0] num2,  input [3:0] num3,  input [3:0] num4,  input [3:0] num5,  input [3:0] num6,  input [3:0] num7,  input [3:0] num8,  output [6:0] show\_num,  output [7:0] show\_port,  input clear  );  wire [6:0] n1, n2, n3, n4, n5, n6, n7, n8;  num2seg ns1 (num1, n1);  num2seg ns2 (num2, n2);  num2seg ns3 (num3, n3);  num2seg ns4 (num4, n4);  num2seg ns5 (num5, n5);  num2seg ns6 (num6, n6);  num2seg ns7 (num7, n7);  num2seg ns8 (num8, n8);  show\_seg ss (clk, n1, n2, n3, n4, n5, n6, n7, n8, show\_num, show\_port, clear);  endmodule |

### 数字信号转化模块num2seg（将数字转化为数码管信号）Verilog代码

|  |
| --- |
| 程序2-7 数字信号转化模块num2seg（将数字转化为数码管信号）Verilog代码 |
| module num2seg(  input [3:0] num,  output reg [6:0] seg  );  always @(num) begin  case (num)  0: seg <= ~'b1111110;  1: seg <= ~'b0110000;  2: seg <= ~'b1101101;  3: seg <= ~'b1111001;  4: seg <= ~'b0110011;  5: seg <= ~'b1011011;  6: seg <= ~'b1011111;  7: seg <= ~'b1110000;  8: seg <= ~'b1111111;  9: seg <= ~'b1111011;  default: seg <= ~'b0000001;  endcase  end  endmodule |

### 数码管显示模块（扫描显示7段数码管）Verilog代码

|  |
| --- |
| 程序2-8 数码管显示模块（扫描显示7段数码管）Verilog代码 |
| module show\_seg(  input clk,  input [6:0] num1,  input [6:0] num2,  input [6:0] num3,  input [6:0] num4,  input [6:0] num5,  input [6:0] num6,  input [6:0] num7,  input [6:0] num8,  output reg [6:0] final\_num,  output reg [7:0] show\_port,  input clear  );    reg [31:0] counter;  initial begin  counter = 0;  end  always @(posedge clk) begin  if (counter == 800000) counter <= 0;  else counter <= counter + 1;  end  always @(\*) begin  if (clear) begin  final\_num = num1;  show\_port = 'b11111111;  end  else if (counter < 100000) begin  final\_num = num1;  show\_port = 'b11111110;  end  else if (counter > 100000 && counter < 200000) begin  final\_num = num2;  show\_port = 'b11111101;  end  else if (counter > 200000 && counter < 300000) begin  final\_num = num3;  show\_port = 'b11111011;  end  else if (counter > 300000 && counter < 400000) begin  final\_num = num4;  show\_port = 'b11110111;  end  else if (counter > 400000 && counter < 500000) begin  final\_num = num5;  show\_port = 'b11101111;  end  else if (counter > 500000 && counter < 600000) begin  final\_num = num6;  show\_port = 'b11011111;  end  else if (counter > 600000 && counter < 700000) begin  final\_num = num7;  show\_port = 'b10111111;  end  else begin  final\_num = num8;  show\_port = 'b01111111;  end  end    endmodule |

## 仿真过程

为了验证设计的正确性，对main、countdown等模块进行了仿真，具体过程如下。

### main模块仿真

由于主模块较为复杂，主模块仿真分为多个部分。

1. 模式切换仿真

在模式切换仿真中，通过choose\_mode输入3个脉冲信号。按照正常的功能，蜂鸣指示灯应该随着脉冲出现3个相同的脉冲，之后mode\_state应从1变为4，且开始执行第4种洗衣模式单漂。仿真代码为程序2-9，实际仿真图如图2-5所示，完全符合预测。

|  |
| --- |
| 程序2-9 主模式模块切换仿真Verilog代码 |
| module main\_sim();  reg reset, clk, pause, choose\_mode, child;  reg [1:0] clo\_weight;  wire bee, power\_on, start, washing, rinsing, drying, water\_in, water\_out;  wire [6:0] show\_num;  wire [3:0] work\_state, mode\_state;  wire hold, cp, final\_mode\_wash, final\_mode\_rinse, final\_mode\_dry;;  wire [1:0] recount, time\_up, weight;  wire [7:0] num1, num2, num3, show\_port;  initial begin  reset <= 1;  clk <= 0;  pause <= 1;  choose\_mode <= 0;  clo\_weight <= 1;  child <= 0;  #1 clo\_weight = 1;  #100 choose\_mode = 1;  #8 choose\_mode = 0;  #8 choose\_mode = 1;  #8 choose\_mode = 0;  #8 choose\_mode = 1;  #8 choose\_mode = 0;  #1 pause = 0;  end  main m (reset, clk, pause, choose\_mode, clo\_weight, child, bee, show\_num, show\_port,  power\_on, start, washing, rinsing, drying, water\_in, water\_out, final\_mode\_wash,  final\_mode\_rinse, final\_mode\_dry, mode\_state,  time\_up, work\_state, hold, recount, cp, num1, num2, num3, weight);  always #1 clk = ~clk;  endmodule |

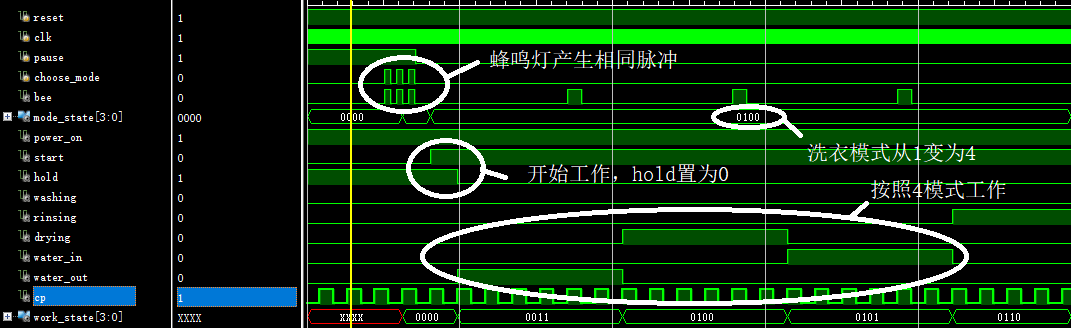


图2-5 main模块模式切换仿真图

1. 完整工作仿真（以模式1洗漂脱为例）

打开开关之后，直接按照默认的模式1洗漂脱工作，预计work\_state将随着步骤的切换而变化，每次步骤切换bee都会收到一个脉冲信号，步骤指示灯将随之跳转，模式指示灯在模式结束后会熄灭。最后在bee收到3个脉冲之后，power\_on会置为0，自动关机。仿真代码为程序2-10，实际仿真图如图2-6所示，完全符合预测。

|  |
| --- |
| 程序2-10 主模块模式切换仿真Verilog代码 |
| module main\_sim();  reg reset, clk, pause, choose\_mode, child;  reg [1:0] clo\_weight;  wire bee, power\_on, start, washing, rinsing, drying, water\_in, water\_out;  wire [6:0] show\_num;  wire [3:0] work\_state, mode\_state;  wire hold, cp, final\_mode\_wash, final\_mode\_rinse, final\_mode\_dry;;  wire [1:0] recount, time\_up, weight;  wire [7:0] num1, num2, num3, show\_port;  initial begin  reset <= 1;  clk <= 0;  pause <= 1;  choose\_mode <= 0;  clo\_weight <= 1;  child <= 0;  #1 clo\_weight = 1;  #100 choose\_mode = 1;  #8 choose\_mode = 0;  #8 choose\_mode = 1;  #8 choose\_mode = 0;  #8 choose\_mode = 1;  #8 choose\_mode = 0;  #1 pause = 0;  end  main m (reset, clk, pause, choose\_mode, clo\_weight, child, bee, show\_num, show\_port,  power\_on, start, washing, rinsing, drying, water\_in, water\_out, final\_mode\_wash,  final\_mode\_rinse, final\_mode\_dry, mode\_state,  time\_up, work\_state, hold, recount, cp, num1, num2, num3, weight);  always #1 clk = ~clk;  endmodule |

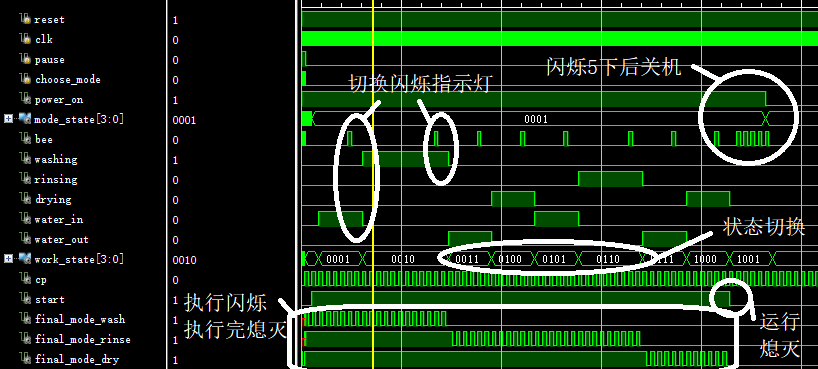


图2-6 main模块完整工作仿真图

1. 中途暂停和切换模式仿真

初始工作模式为模式2单洗，在暂停之后，输入2个choose\_mode的脉冲，下一次启动时应变为模式4单漂。在暂停期间，洗衣机应不继续工作，直到pause变为0。仿真代码为程序2-11，实际仿真图如图2-7所示，完全符合预测。

|  |
| --- |
| 程序2-11 主模块中途暂停和切换模式仿真Verilog代码 |
| module main\_sim();  reg reset, clk, pause, choose\_mode, child;  reg [1:0] clo\_weight;  wire bee, power\_on, start, washing, rinsing, drying, water\_in, water\_out;  wire [6:0] show\_num;  wire [3:0] work\_state, mode\_state;  wire hold, cp, final\_mode\_wash, final\_mode\_rinse, final\_mode\_dry;;  wire [1:0] recount, time\_up, weight;  wire [7:0] num1, num2, num3, show\_port;  initial begin  reset <= 1;  pause <= 1;  clk <= 0;  choose\_mode <= 0;  clo\_weight <= 1;  child <= 0;  #2 choose\_mode <= 1;#2 choose\_mode <= 0;  #1 pause = 0;  #500 pause = 1;  #20 choose\_mode <= 1;#50 choose\_mode <= 0;  #20 choose\_mode <= 1;#50 choose\_mode <= 0;  #500 pause <= 0;  end  main m (reset, clk, pause, choose\_mode, clo\_weight, child, bee, show\_num, show\_port,  power\_on, start, washing, rinsing, drying, water\_in, water\_out, final\_mode\_wash,  final\_mode\_rinse, final\_mode\_dry, mode\_state,  time\_up, work\_state, hold, recount, cp, num1, num2, num3, weight);  always #1 clk = ~clk;  endmodule |

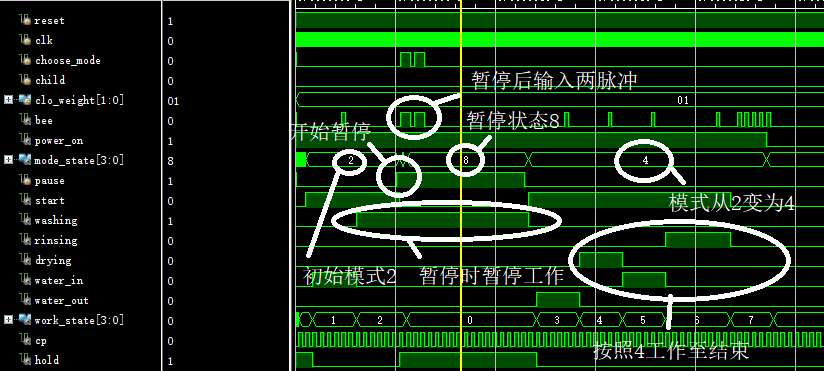


图2-7 main模块中途暂停和切换模式仿真图

1. 其它附加功能仿真

在上述的仿真中，还有一些小的功能未能覆盖，包括重量改变、儿童锁、防误触、预约功能。其中由于预约功能耗时较长，不适合仿真。因此本部分主要针对前三个功能。

暂停时改变重量时，num1~num3都应随之改变。运行时，保持儿童锁关闭，在一个周期内打开和关闭开关，洗衣机仍能正常工作。当打开儿童锁后，将开关关闭3s左右，再打开，洗衣机仍能正常工作。当关闭儿童锁后，若在将开关关闭1s以上，则洗衣机会停止运行。仿真代码为程序2-12，实际仿真图如图2-8所示，完全符合预测。

|  |
| --- |
| 程序2-12 主模块防误触、儿童锁、重量设置仿真Verilog代码 |
| module main\_sim();  reg reset, clk, pause, choose\_mode, child;  reg [1:0] clo\_weight;  wire bee, power\_on, start, washing, rinsing, drying, water\_in, water\_out;  wire [6:0] show\_num;  wire [3:0] work\_state, mode\_state;  wire hold, cp, final\_mode\_wash, final\_mode\_rinse, final\_mode\_dry;  wire [1:0] recount, time\_up, weight;  wire [7:0] num1, num2, num3, show\_port;  initial begin  reset = 1;  pause = 1;  clk = 0;  choose\_mode = 0;  child = 0;  #2 choose\_mode = 1;  #2 choose\_mode = 0;  #50 clo\_weight = 'b11;  #50 clo\_weight = 'b10;  #50 clo\_weight = 'b01;  #50 clo\_weight = 'b00;  #1 pause = 0;  #50 reset = 0;  #5 reset = 1;  #50 child = 1;  #10 reset = 0;  #200 reset = 1;  #10 child = 0;  #100 reset = 0;  end  main m (reset, clk, pause, choose\_mode, clo\_weight, child, bee, show\_num, show\_port,  power\_on, start, washing, rinsing, drying, water\_in, water\_out, final\_mode\_wash,  final\_mode\_rinse, final\_mode\_dry, mode\_state,  time\_up, work\_state, hold, recount, cp, num1, num2, num3, weight);  always #1 clk = ~clk;  endmodule |

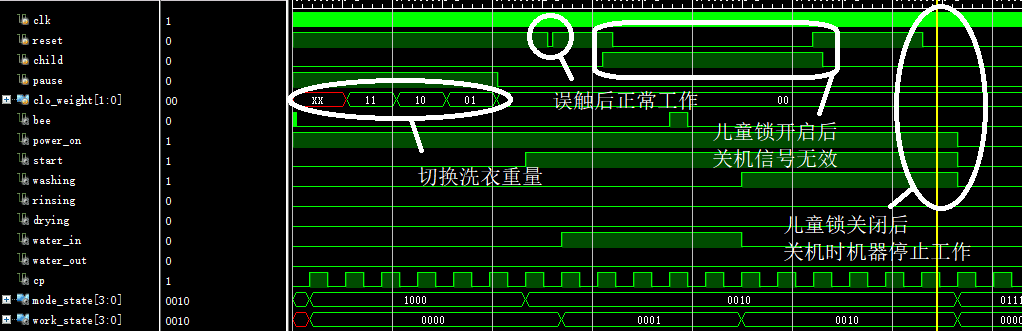


图2-8 main模块其他附加功能仿真图

### countdown模块仿真

countdown模块负责实现倒计时功能。在reset=’b00的时候开始倒计时，否则重置数字，计时完成后对应的time\_up位会置1。仿真代码为程序2-13，实际仿真图如图2-9所示，完全符合预测。

|  |
| --- |
| 程序2-13 countdown模块倒计时仿真Verilog代码 |
| module countdown\_sim();  reg clk, hold\_wire;  reg [1:0] reset;  reg [7:0] num1, num2, num3;  wire [7:0] w1, w2, w3;  wire [1:0] zero;  wire [1:0] time\_up;  wire [6:0] show\_num;  wire [7:0] show\_port;  wire cp;  reg clear;  divider d (clk, cp);  countdown cd1 (clk, cp, reset, hold\_wire, num1, num2, num3, time\_up,  show\_num, show\_port, clear, w1, w2, w3);  always #1 clk = ~clk;  initial begin  num1 = 12;  num2 = 15;  num3 = 49;  clk = 0;  reset = 'b10;  hold\_wire = 0;  clear = 0;  #1000 reset = 'b11;  #1000 reset = 'b00;  #1000 reset = 'b01;  end  endmodule |

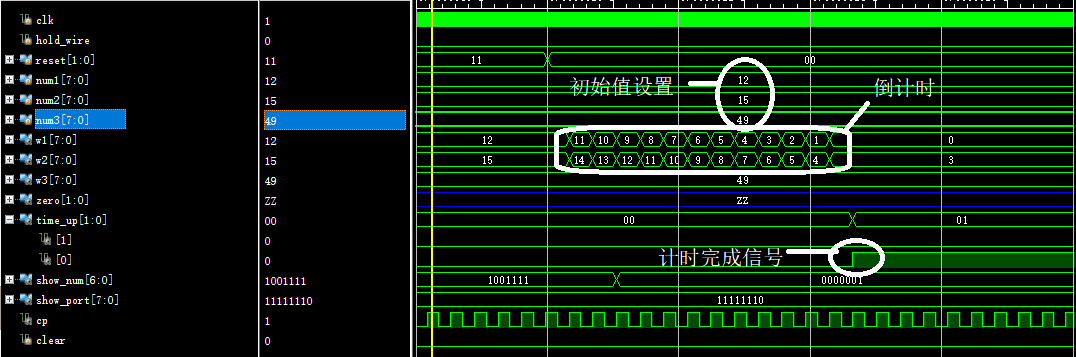


图2-9 countdown倒计时模块仿真图

## 主要故障

### 故障1

**问题描述：**Vivado报告warning: inferring latch for variable ‘var’，最后前仿真正确但后仿真出现错误。

**问题分析：**经查阅官方文档，原因为在编写组合电路时，若一个变量在if中赋值而在else中未赋值，则会弹出这个警告。因为这种写法在生成电路时会产生锁存器，可能会带来不可预知的错误。

**解决方法：**在每一个if-else和case语句中，使用{washing, rinsing, drying, water\_in, water\_out} = 'b00000;类似的赋值语句，保证每一个变量都被成功的赋值，而不会产生锁存器。修改后的代码的一部分如程序2-14所示

|  |
| --- |
| 程序2-14 case语句中部分修改的语句块 |
| 1: begin //water in  {washing, rinsing, drying, water\_in, water\_out} = 'b00010;  {mode\_wash, mode\_rinse, mode\_dry} = 'b111;  hold = 0;  end\_reset = 0;  if (time\_up[1]) begin  next\_work\_state = 2;  num2 = 9;  recount = 'b10;  end  else begin  next\_work\_state = 1;  num2 = weight;  recount = 'b00;  end  end  2: begin //washing  {washing, rinsing, drying, water\_in, water\_out} = 'b10000;  {mode\_wash, mode\_rinse, mode\_dry} = 'b111;  hold = 0;  end\_reset = 0;  if (time\_up[1]) begin  next\_work\_state = 3;  num2 = weight;  recount = 'b10;  end  else begin  next\_work\_state = 2;  num2 = 9;  recount = 'b00;  end  end |

### 故障2

**问题描述：**仿真时发现倒计时与灯的变化不同步。

**问题分析：**经分析代码，发现在main与countdown之间传参数时只将原始的clk传入countdown后，在countdown中进行分频，控制倒计时。很可能是由于分频的不一致导致了出现误差。

**解决方法：**使用统一的时钟，除了将显示数码管所必须的clk原始时钟传入以外，还需要传入主模块中分频好的时钟信号cp，保证控制的是同一个时钟。进行改写后，问题解决，两个时钟同步。修改后的countdown模块声明部分代码如程序2-15所示。

|  |
| --- |
| 程序2-15 时钟同步修改后的countdown声明部分 |
| module countdown(  input clk\_k, // original 100MHz  input clk, // 1Hz clock  input [1:0] reset, // reset[0] : first number , reset[1] : second number  input hold\_wire, // 1~3 : 3 numbers hold  input [7:0] num1,  input [7:0] num2,  input [7:0] num3, // constant  output reg [1:0] final\_time\_up,  output [6:0] show\_num,  output [7:0] show\_port,  input clear  //debug  ,output w1,  output w2,  output w3  ); |

### 故障3

**问题描述：**在编写状态机的过程中，出现multi-net的warning，最终生成比特流失败。

**问题分析：**经查阅文献发现，当在多个always块中修改同一个寄存器的值，会导致该warning，原因是电路中不可能将两根线直接相连。

**解决办法：**添加一个reset寄存器，另一个always块需要修改var的值时，将reset置1。原始的always则在处理的时候加上reset的判断。如程序2-16所示，最终成功编写出对应功能的电路。

|  |
| --- |
| 程序2-16 带有reset的flag处理电路 |
| always @(posedge clk) begin  if (flag\_reset) begin  // reset  flag <= 0;  end  else if ((mode\_state == 'b0000) & choose\_mode) begin  flag <= 1;  end  end |

## 功能测试

共进行了2项测试，它们分别为：正常运行与切换测试，异常操作测试。

输入包括左边的两个开关，从左至右分别代表儿童锁和预约10秒；右边的四个开关，从左至右分别代表水位开关1、水位开关2、暂停开关、电源开关；P17按钮，代表模式切换。

输出包括8个7段数码管，显示3个2位数，从左至右分别代表总用时、本阶段时间、进水量（衣物重量）；左边三个灯，分别代表脱水模式、漂洗模式、洗涤模式；右边8个灯，0灯代表电源指示，1灯代表运行指示，2灯代表蜂鸣指示，3-7代表5个洗衣过程。

### 正常运行与切换测试

开启电源，关闭暂停，将重量设置为5，开始正常运行直到结束自动关机。

首先将重量控制开关预置为5，暂停开关拨到0，打开电源开关，观察自动运行结果。刚打开时，显示如图2-10所示。此时6号灯亮，表示正在脱水，13、14、15号灯均亮，表示3个部分都将要运行，其中15号灯闪烁，表示正在进行洗衣部分。

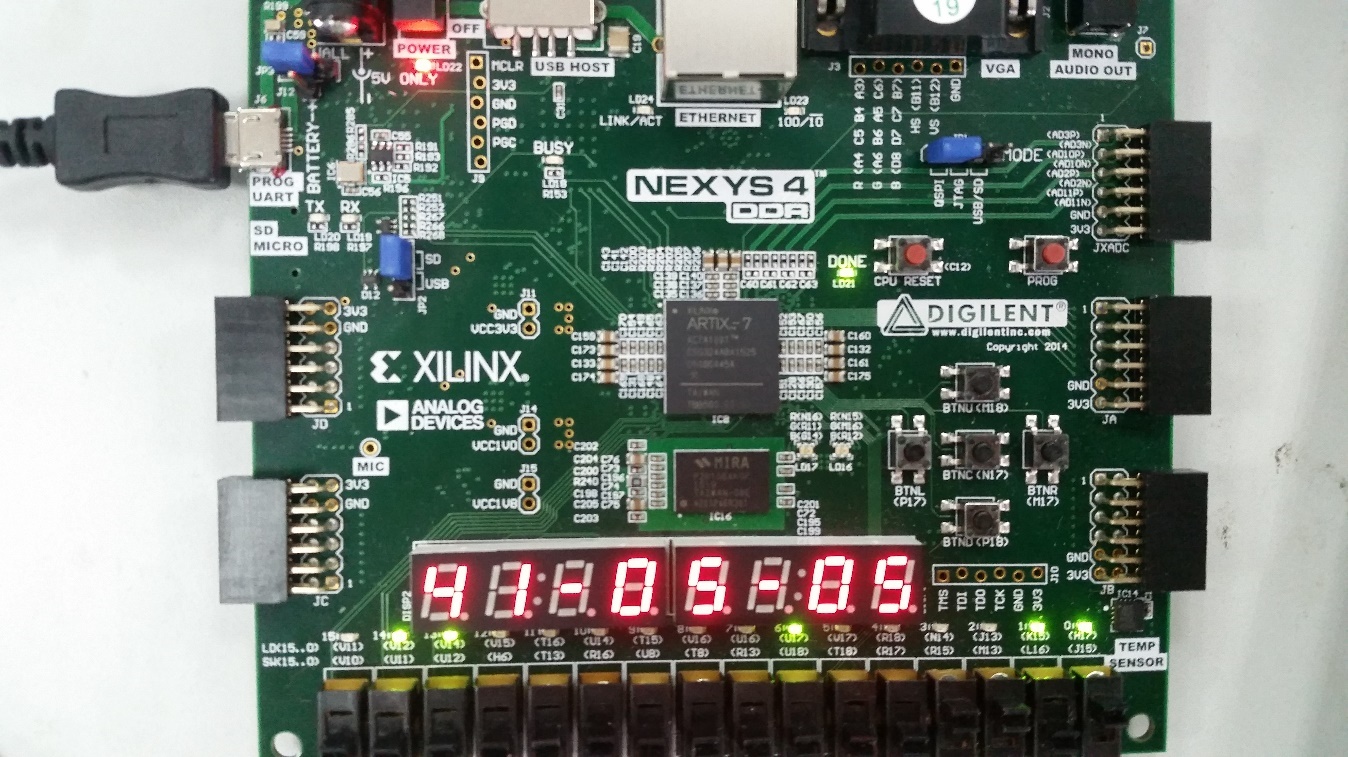


图2-10 正常运行与切换测试启动图

继续运行，第二个数字变为0时，跳到下一个过程洗涤，但仍属于洗涤状态，如图2-11所示。3号灯亮，表示正在进行洗涤步骤，13、14、15号灯仍然均亮，表示3个部分都将要运行，其中15号灯闪烁，表示正在进行洗衣部分。由于洗衣部分包含进水与洗衣，因此测试正常。



图2-11 正常运行与切换测试下一步骤

继续运行，倒计时2再次为0时，预计变为下一模式，开始排水过程，实际结果如图2-12所示。6灯亮，表示正在排水，15灯熄灭，表示洗涤部分已经完成，而13、14灯亮，其中14灯闪烁，表示正在进行漂洗部分。切换过程中蜂鸣灯闪一下，验证完全符合预期。

此时打开暂停开关，发现停止倒计时，此时按动按钮切换状态，切换之后应显示切换后的时间和切换后的状态灯。实际切换后如图2-13所示，每次按动按钮时，蜂鸣灯都会亮，14、15灯亮，对应洗漂模式，而洗漂模式的时间为33分钟，其中第一过程进水共需要5分钟，与显示相同，验证符合要求。

此时通过拨动水位控制开关为2，应控制数码管数字的显示，实际操作如图2-14所示。数码管三个数字都发生了变化，变为24、02、02，与预计相符。

关闭暂停开关，继续运行，发现模式均正常跳变。之后再次打开暂停开关，发现计时停止，洗衣机暂停工作。接着关闭暂停开关，发现洗衣机接着原来的倒计时运行，符合预期结果。

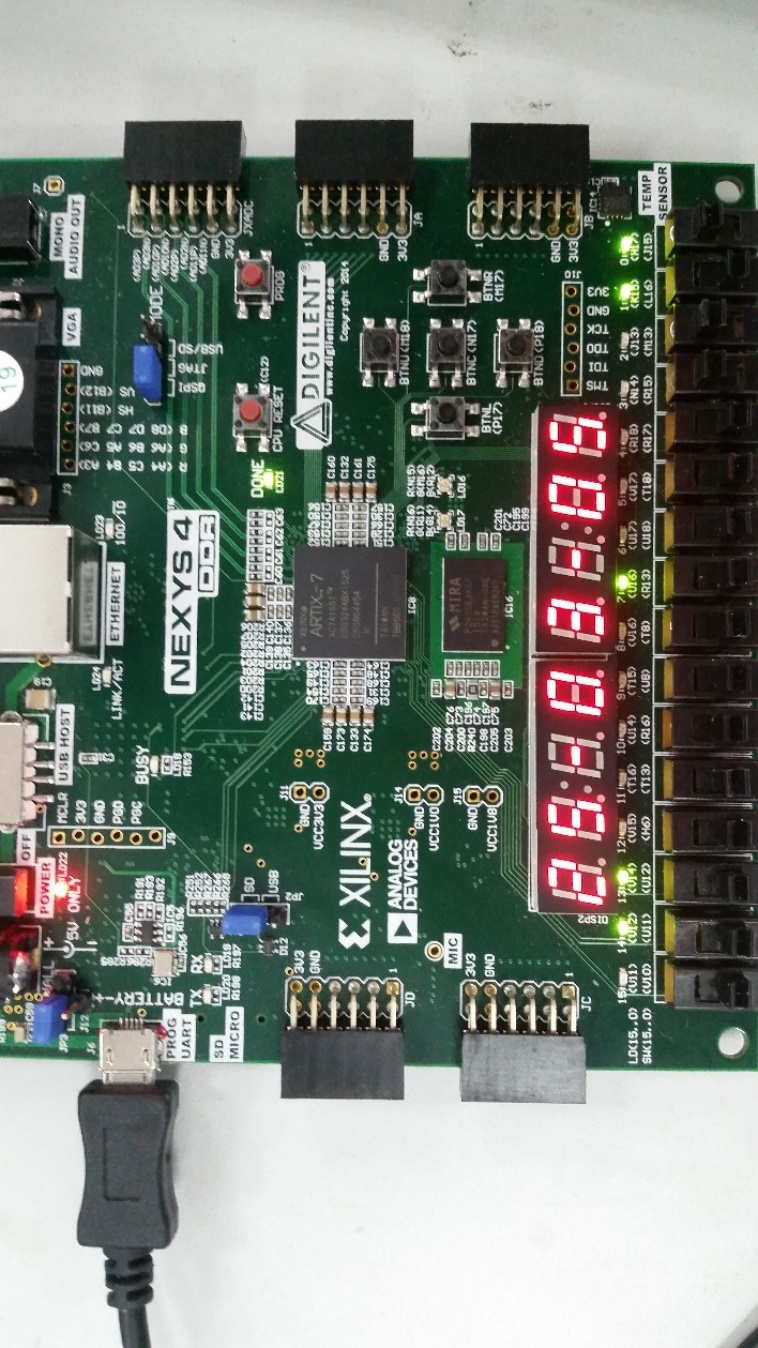


图2-12 正常运行与切换测试下一部分

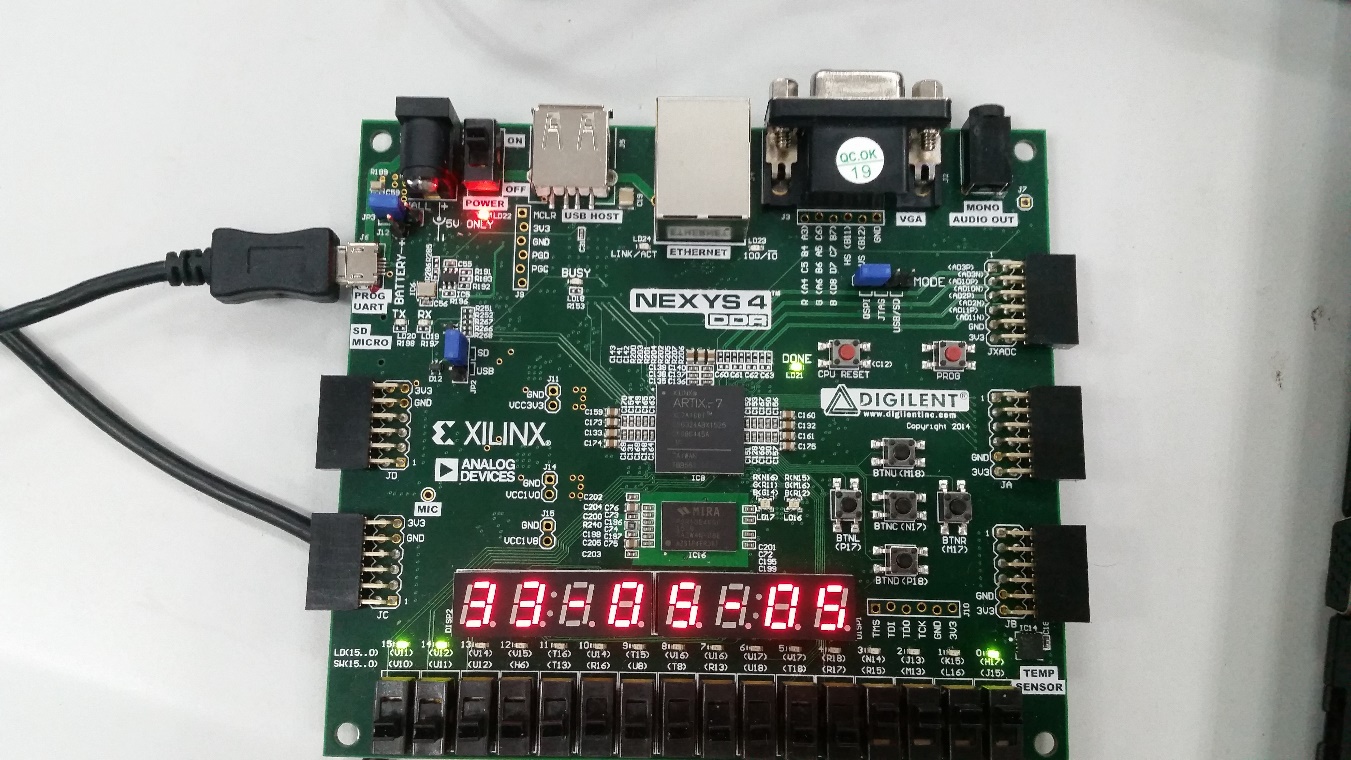


图2-13 正常运行与切换测试切换模式

继续运行至最后一个模式，当数码管第1、2个数字均变为00时，运行灯和其他状态灯均熄灭，只有电源灯长亮，蜂鸣灯闪烁，如图2-15所示。最后闪烁5下后，所有灯熄灭，数码管显示关闭，机器关闭，如图2-16所示，符合预期。

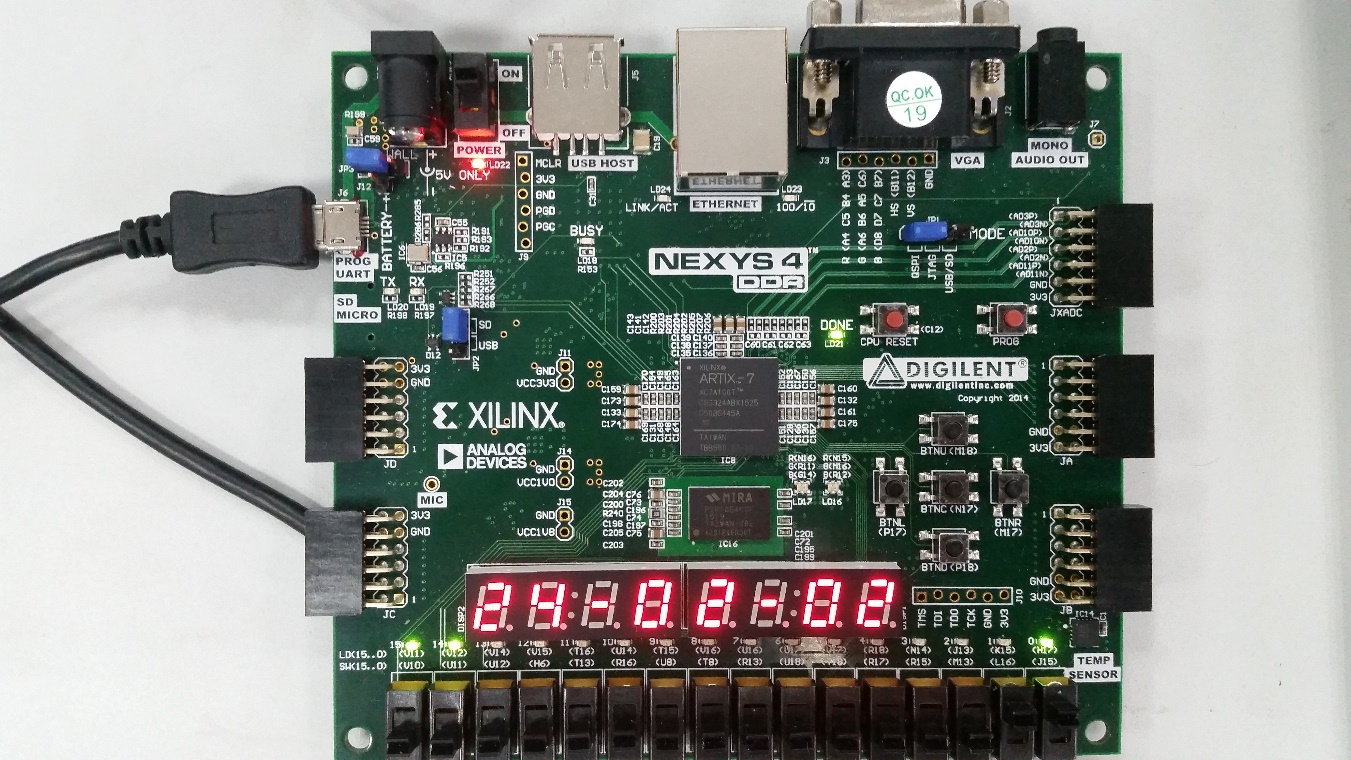


图2-14 正常运行与切换测试改变水位

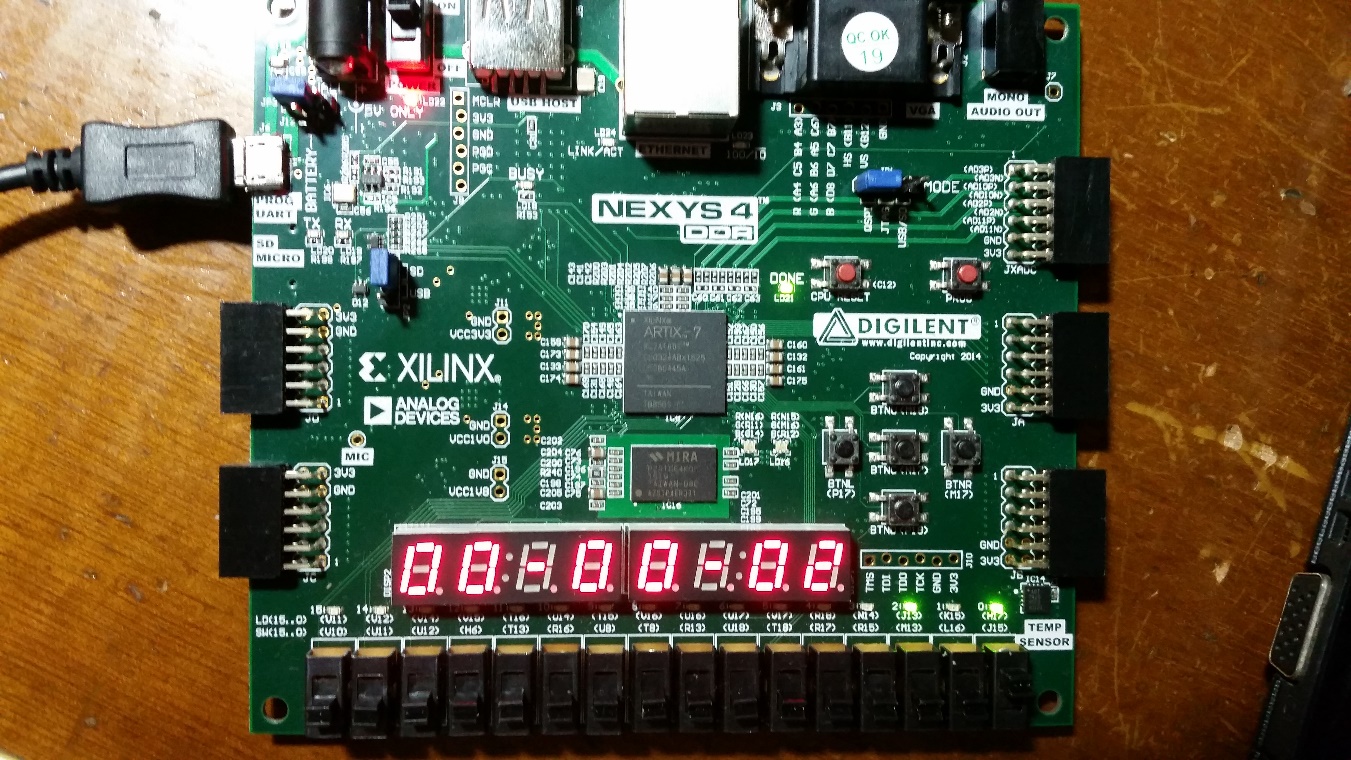


图2-15 正常运行与切换测试结束前夕

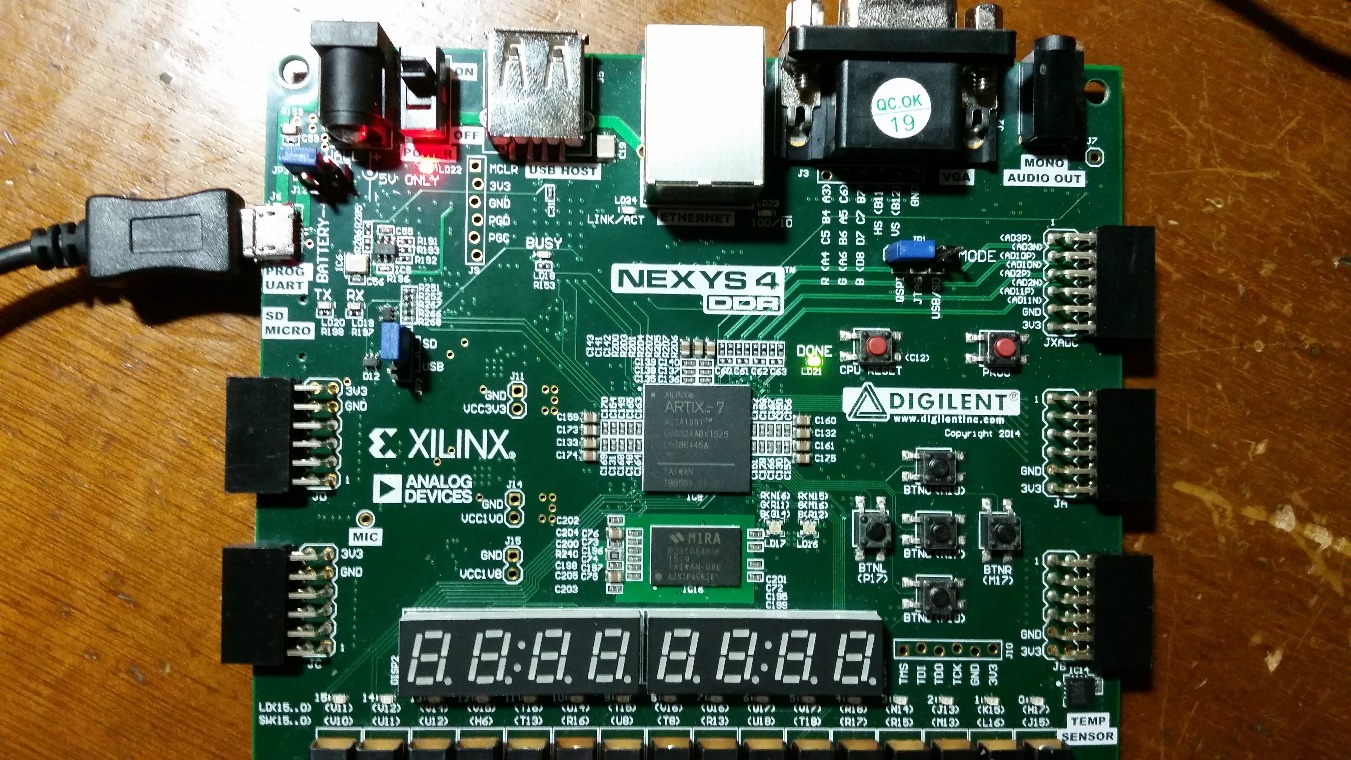


图2-16 正常运行与切换测试自动关机之后

### 异常操作测试

首先保持关机模式，拨动暂停开关，按动按钮均无反应，符合预期。开机正常运行，在正在执行的过程中连续按下按钮，只有蜂鸣灯亮起，但正常运行不受影响，如图2-17所示，符合预期。之后，在仍在运行中时，关闭电源开关，则所有灯熄灭，数码管不显示，如图2-18所示，最后尝试按动按钮和波动其他开关，发现同样没有反应。以上异常测试没有发现错误，顺利通过。

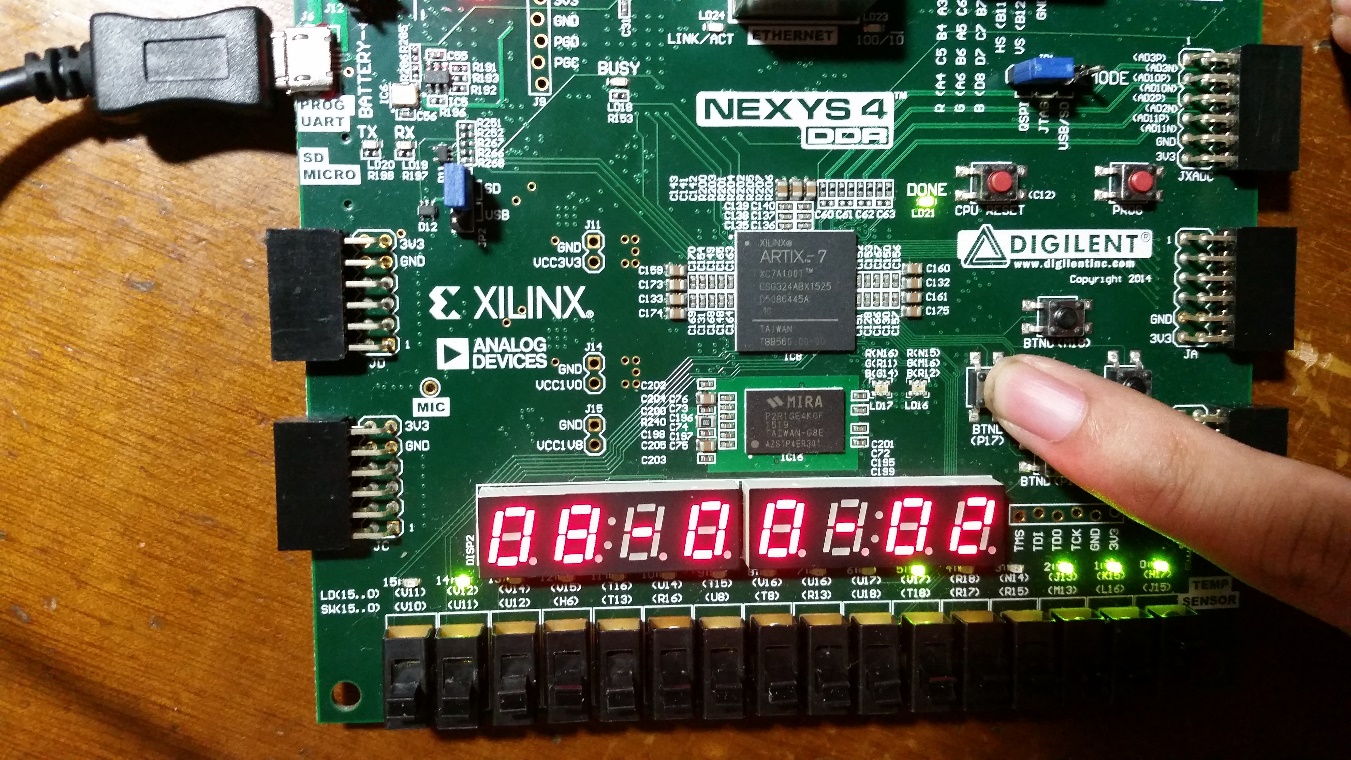


图2-17 异常操作测试运行中按切换模式按钮

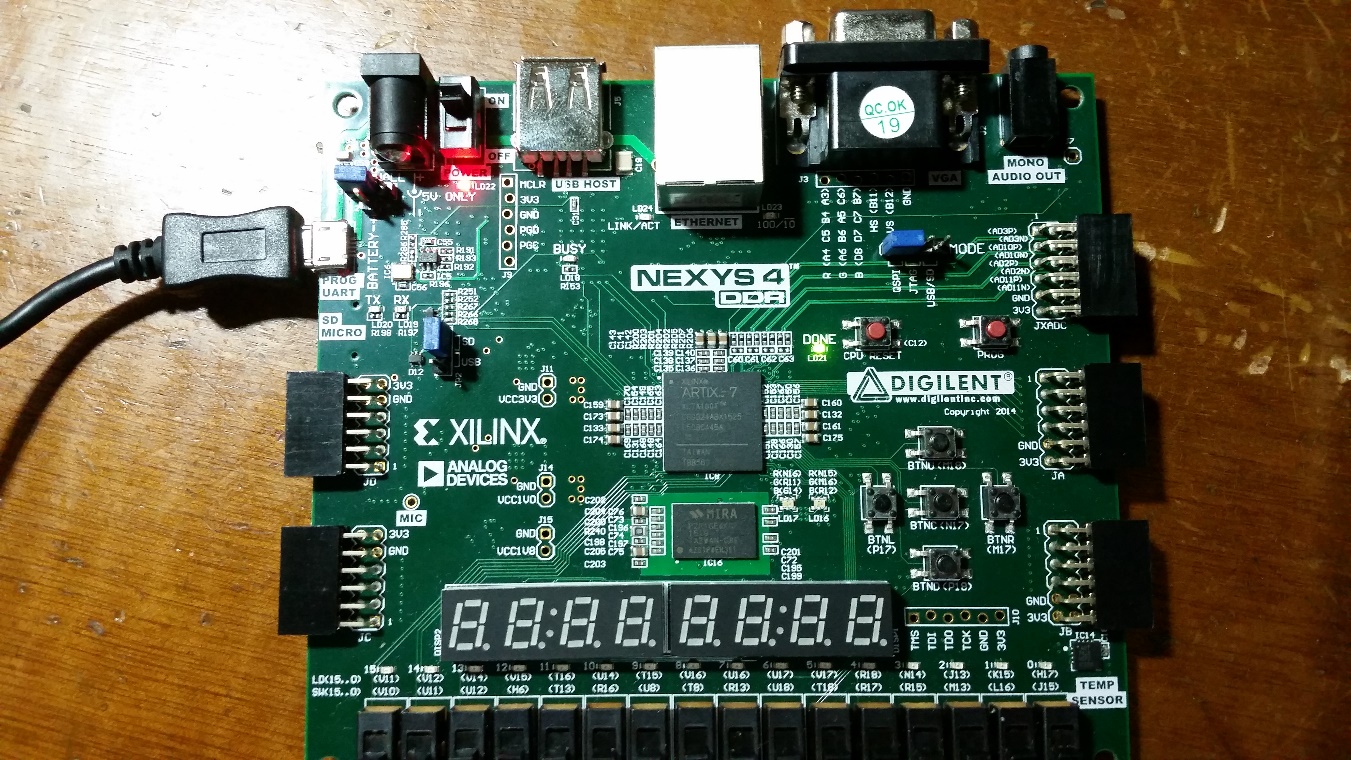


图2-18 异常操作测试强制关闭

## 实验中遇到的主要问题及解决方法

表3-1 主要问题及解决方法

|  |  |
| --- | --- |
| 问题 | 解决方案 |
| 如何解决每一个模式灯亮、闪烁、熄灭的切换问题 | 将灯的灭和亮视为两个状态，在每一个工作状态与运行状态中均对三个模式灯进行赋值，保证每一次模式变化的时候灯的亮和灭正确。由于无论是哪一种洗衣模式，洗漂脱的顺序都是不变的。于是编写一个always块，使每次在运行的时候依次判断洗、漂、脱是否亮，如果亮，则将其和1Hz时钟cp进行与运算，得到输出结果。 |
| 如何在8个7段数码管上显示不同的数字 | 通过1kHz的时钟信号对8个数码管进行扫描。依次循环地对8个数码管进行赋值，当频率达到1kHz左右时，可以达到稳定显示的目的。 |
| 如何解决暂停状态既可以切换洗衣模式又可以继续计时的功能 | 增加一个预暂停工作状态。每次暂停时会进入预暂停状态，此时如果暂停开关关闭，则回到原来的工作状态。但如果此时按下按钮，则将在下一个时钟跳到暂停状态，同时目标状态也会切到下一个。在暂停状态，工作时间初始化，状态随按钮的指示变化。 |
| 如何实现自动关机功能 | 在每次到达一个洗衣模式的最后结束（default）工作状态时，触发一个倒计时，当倒计时输出时间到的信号时，将强制停止寄存器置为1，工作模式跳转为停止，即电源断电。同时在电源开启状态时，要判断强制停止寄存器是否为1，如果为1，则必须要关闭一次再开启才能重新启动。 |
| 如何实现蜂鸣器指示灯的多功能显示 | 蜂鸣器指示灯的显示与time\_up（时间到）、sel\_mode（按钮信号）、cp（时钟）有关。在暂停时间，time\_up一定为0，此时sel\_mode与蜂鸣指示灯直接相连；在模式切换时间，time\_up有一个1的脉冲；在结束时间，time\_up会一直保持1，此时将time\_up&cp就可以达到蜂鸣灯闪烁的效果。 |

## 实验方案的改进意见

### 儿童锁

（1）设计需求

在洗衣过程中，很可能由于儿童的误操作导致洗衣过程终止，造成了不必要的麻烦。在这样的背景下，设计了儿童锁功能。打开儿童锁开关后，洗衣机将保持原先的工作状态，不受电源等开关的控制，直到洗衣过程结束或关闭儿童锁按钮。

（2）改进方案

a. 具体改进的模块结构设计

由于设计上使用状态机控制，实际并不需要单独设置儿童锁模块，只需要在每一个通过reset跳转到关闭的条件处加上child==0的条件，即可达到儿童锁打开时不能关机的目的，pause同理。

b. 具体改进的模块Verilog代码

具体改进的Verilog代码分散在状态机的各个部分，主要为加一个限制条件，其中一部分代码如程序2-17所示。

|  |
| --- |
| 程序2-17 儿童锁增加部分代码示例 |
| /////////////////////////// changed part ////////////////////////////  else if (reset == 0 && child == 0) begin  next\_mode\_state = 7;  next\_work\_state = 0;  end\_reset = 1;  end  //////////////////////// change part ///////////////////////////////// |

（3）仿真

图2-19显示了多个功能的仿真，reset为电源信号，child为儿童锁信号。其中，当儿童锁打开（置为1）时，长时间的关机信号不起作用。之后儿童锁关闭（置为0），电源关机信号导致所有指示灯立即置为1。

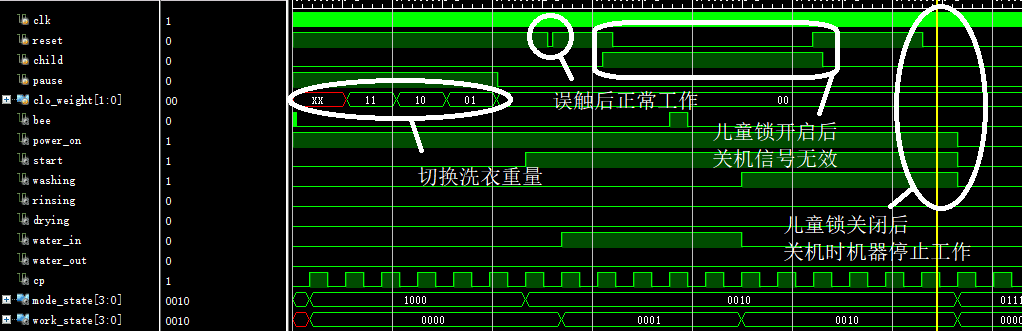


图2-19 儿童锁仿真结果

（4）修改后的功能测试

打开开关，使其正常运行，之后打开儿童锁，然后关闭电源，电路继续运行不受影响，如图2-20所示。之后关闭儿童锁，由于此时电源开关仍处于关闭状态，电路立即停止工作。此次测试符合预期结果。



图2-20 打开儿童锁后关机

### 防误触

1. 设计需求

在实际操作过程中，有时会由于误按按钮导致机器断电，只能重新开始洗衣过程，造成了不必要的麻烦。在这样的背景下，设计了防误触功能。在关闭电源开关之后在1秒内打开电路仍可以照常继续工作，但若超过1秒则设备关闭。

1. 改进方案
2. 具体改进的模块结构设计

防误触是在整个系统的设计阶段就已经确定的，由于系统采用1秒为周期的Moore型电路，因此reset输入在下一个时钟到来之前不会起到作用。只需要保证更新的周期，就可以保证在1秒内的迅速变化不会引起洗衣机的停机。

1. 具体改进的模块Verilog代码

具体改进的Verilog代码为状态的切换控制部分，如程序2-18所示。

|  |
| --- |
| 程序2-18 状态切换控制代码 |
| divider d (clk, cp);  always @(posedge cp) begin  if (reset == 0 && child == 0) begin  // reset  last\_mode\_state <= 7;  mode\_state <= 7;  end  else begin  mode\_state <= next\_mode\_state;  end  end  always @(posedge cp) begin  if (reset == 0 && child == 0) begin  // reset  work\_state <= 0;  last\_work\_state <= 0;  end  else begin  work\_state <= next\_work\_state;  end  end |

1. 仿真

图2-20显示了多个功能的仿真，reset在开始的时候又一次快速的置0再置1，此时child儿童锁始终为0，但电源没有关闭，符合防误触要求。

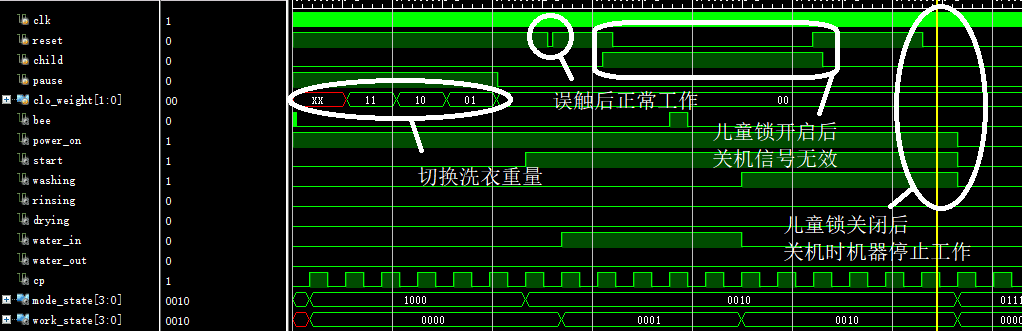


图2-20 防误触仿真结果

1. 修改后的功能测试

打开开关，使洗衣机正常运行，保持儿童锁出于关闭状态。快速的开关reset按钮多次，发现均无影响，仍然可以继续运行。最后将reset置为0，约1秒后发现系统停止工作，显示消失。此次功能测试完全符合预期。

### 预约

（1）设计需求

在实际使用过程中，常常由于无法准时操作，需要预约一定时间后自动开启洗衣。于是设计了预约功能，通过若干个开关控制预约时间后，洗衣机将在预约时间后开始运行。

（2）改进方案

1. 具体改进的模块结构设计

增加一个预约状态，每次进入的时候首先进入预约状态，如果没有预约，则跳到初始暂停状态；如果有预约，则调用倒计时模块开始预约倒计时，收到计时结束的信号之后，再跳到初始状态，开始正常执行。

1. 具体改进的模块Verilog代码

具体的改进主要为在一级状态机中加入一个新的状态预约状态。

|  |
| --- |
| 程序2-19 预约状态代码段 |
| 9： begin  {power\_on, start} = 'b10;  if (reserve) begin  num1 = 5; // reserve for 5 sec  num2 = 5; // reserve for 5 sec  num3 = 5; // reserve for 5 sec  if (time\_up[0] | time\_up[1] == 1) begin  next\_mode\_state = 8;  end  end  else begin // not reserved  next\_mode\_state = 8;  end  end |

（3）仿真

由于预约模块设定为秒为单位，不便于进行仿真，因此直接在开发板上实现。

（4）修改后的功能测试

修改之后，首先将预约按钮打开，之后开始倒计时，1、2个数码管显示剩余等待时间，3数码管显示总预约时间5秒，如图2-21所示，之后时间到之后，进行正常的运行功能。本次设计完全符合预期。

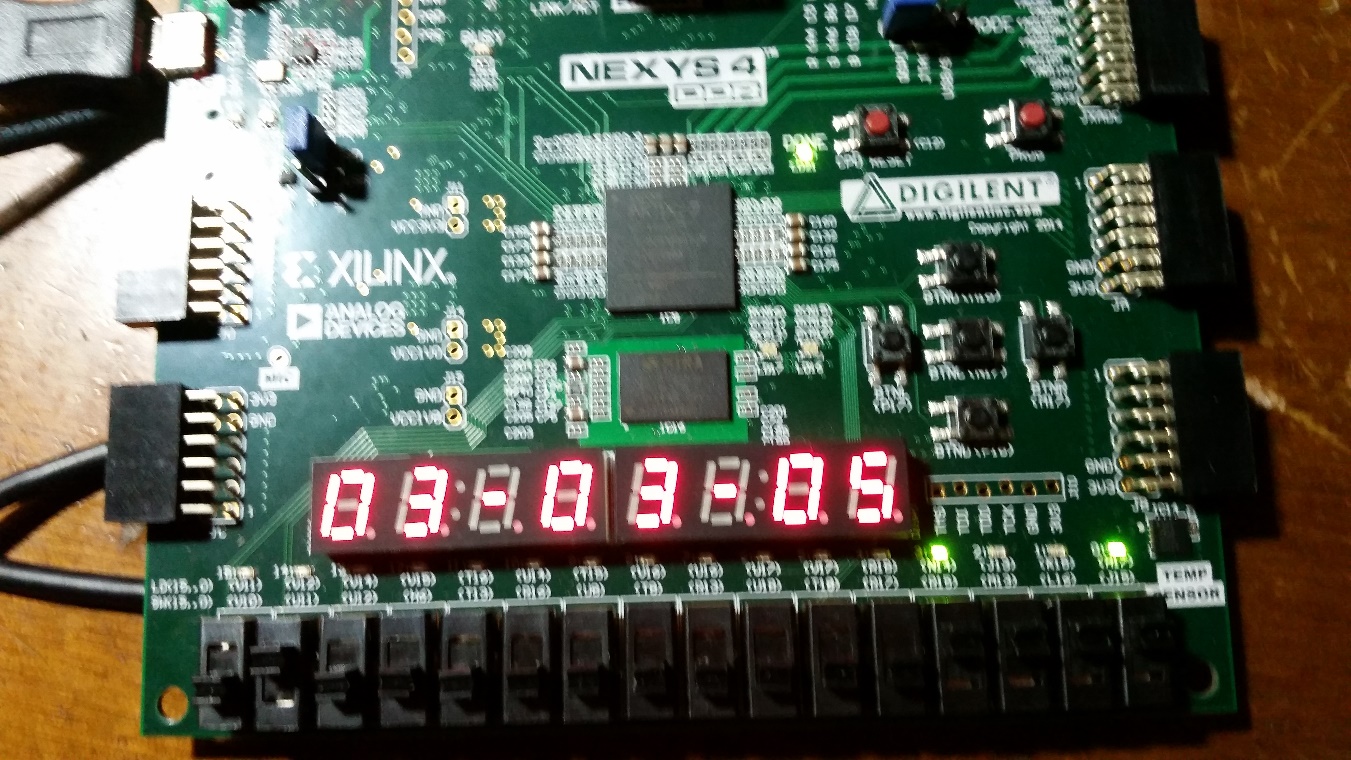


图2-21 预约倒计时过程

# 总结与心得

## 课设总结

在洗衣机的设计过程中，主要完成了以下工作。

（1）前期对于代码架构的分析和模块分布的设计，包括绘制状态图，设置有限状态机。

（2）初步开始时对辅助模块的编写，包括倒计时模块，分频器模块等，并仿真和使用开发板验证其正确性。

（3）中期对主模块的循序渐进的编写，先实现一个洗衣模式的功能，再将其扩展到其它5个洗衣模式。

（4）中后期开始编写模式切换，将各个模式连接起来，同时加入了暂停、开关等处理。

（5）后期检查程序问题，调试bug，添加亮点功能。

## 课设心得

（1）在选题之前一定要仔细分析不同题目的特点，在有基本的思路之后在开始设计编写程序，选题之后的更改可能会浪费大量的时间。

（2）调试的过程中一定要先进行前后仿真通过后，再在开发板上尽心测试，Implementation和Generate Bitstream会耗费大量的时间。

（3）编写时需注意硬件编程的特殊性。并且需要时刻注意所编写的电路为组合逻辑电路还是时序逻辑电路，注意两种电路不同的编写方法和功能，否则很可能出现逻辑正确但无法综合电路的情况。

（4）注意reg寄存器型和wire线网型在电路中不同的使用场合。

（5）尽量更多的分模块，并在各个模块仿真、开发板验证通过后，才开始连接模块。这样既可以增加今后代码的可维护性，又可以使结构更清晰，更容易调试。

# 参考文献

**教学参考书：**

[1]欧阳星明,于俊清. 数字逻辑. 武汉:华中科技大学出版社，2012

[2]白中英,谢松云. 数字逻辑. 北京:科学出版社，2013

[3]徐光辉,程东旭，黄如. 基于FPGA的嵌入式开发与应用. 北京:电子工业出版社, 2006.

[4]Guy Even. Digital Logic Design: A Rigorous Approach, Cambridge University Press, 2012

[5]David Harris. Digital Design and Computer Architecture(Second Edition), Morgan Kaufmann, 2012

[6]Stephen Brown. 数字逻辑基础与Verilog设计, 机械工业出版社, 2009

**课外文献阅读 ：**

[1]Introductory Digital Systems Laboratory

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-111-introductory-digital-systems-laboratory-fall-2002

[2]Digital Design Fabrication

http://ocw.mit.edu/courses/architecture/4-510-digital-design-fabrication-fall-2008/

[3]Analysis and Design of Digital Integrated Circuits

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-374-analysis-and-design-of-digital-integrated-circuits-fall-2003/

# **附 录1（源程序）**

//////////////////////////////////////////////////////////// main.v /////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/08 23:52:33

// Design Name:

// Module Name: main

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module main(

//inputs

input reset,

input clk,

input pause,

input choose\_mode, // button

input [1:0] clo\_weight,

input child,

//outputs

output reg bee,

output [6:0] show\_num,

output [7:0] show\_port,

output reg power\_on,

output reg start,

output reg washing,

output reg rinsing,

output reg drying,

output reg water\_in,

output reg water\_out,

output reg final\_mode\_wash,

output reg final\_mode\_rinse,

output reg final\_mode\_dry

//debug

/\*

,output [3:0] mode\_state,

output time\_up,

output [3:0] work\_state,

output hold,

output recount,

output cp,

output [7:0] num1,

output [7:0] num2,

output [7:0] num3,

output [1:0] weight\*/

);

// 1~6 : working mode, 0 : pausing mode, 7 : power off

reg [3:0] last\_mode\_state;

reg [3:0] mode\_state;

reg [3:0] next\_mode\_state;

wire [3:0] next\_mode;

// work state

reg [3:0] last\_work\_state;

reg [3:0] work\_state;

reg [3:0] next\_work\_state;

//countdown inputs and outputs

reg [1:0] recount;

reg hold; // pause counting

reg [7:0] num1, num2, num3;

wire [1:0] time\_up; // 1st number and 2nd number time\_up

reg [2:0] weight; // the weight of clothes, also water in amount

reg stop;

reg flag\_reset;

reg flag;

reg [1:0] end\_count;

reg [3:0] end\_state;

reg end\_reset, ending;

reg force\_end;

wire clear;

reg clear\_reg;

reg mode\_wash, mode\_rinse, mode\_dry;

initial begin

bee = 0;

mode\_state = 0;

power\_on = 0;

start = 0;

washing = 0;

rinsing = 0;

drying = 0;

water\_in = 0;

water\_out = 0;

//countdown inputs and outputs

recount = 0;

hold = 0;

num1 = 0;

num2 = 0;

num3 = 0;

bee = 0;

stop = 0;

//weight of clothes

weight = 3;

flag = 0;

flag\_reset = 0;

end\_count = 0;

end\_state = 0;

end\_reset = 0;

ending = 0;

force\_end = 0;

clear\_reg = 0;

end

divider d (clk, cp);

countdown cd (clk, cp, recount, hold, num1, num2, num3, time\_up, show\_num, show\_port, clear);

sel\_mode cm (choose\_mode, next\_mode);

always @(posedge clk) begin

if (mode\_wash == 1) begin

final\_mode\_wash = mode\_wash & cp;

final\_mode\_rinse = mode\_rinse;

final\_mode\_dry = mode\_dry;

end

else begin

if (mode\_rinse == 1) begin

final\_mode\_wash = mode\_wash;

final\_mode\_rinse = mode\_rinse & cp;

final\_mode\_dry = mode\_dry;

end

else begin

final\_mode\_wash = mode\_wash;

final\_mode\_rinse = mode\_rinse;

final\_mode\_dry = mode\_dry & cp;

end

end

end

assign clear = clear\_reg;

always @(\*) begin

bee = choose\_mode | ((time\_up[0] | time\_up[1]) & cp);

end

always @(posedge cp or posedge end\_reset) begin

if (end\_reset) begin

// reset

end\_count <= 0;

force\_end <= 0;

end

else if (end\_count == 3) begin

force\_end <= 1;

end

else if (ending == 1) begin

end\_count = end\_count + 1;

end

end

always @(posedge clk) begin

if (flag\_reset) begin

// reset

flag <= 0;

end

else if ((mode\_state == 'b0000) & choose\_mode) begin

flag <= 1;

end

end

always @(posedge cp) begin

if (reset == 0 && child == 0) begin

// reset

last\_mode\_state <= 7;

mode\_state <= 7;

end

else begin

mode\_state <= next\_mode\_state;

end

end

always @(posedge cp) begin

if (reset == 0 && child == 0) begin

// reset

work\_state <= 0;

last\_work\_state <= 0;

end

else begin

work\_state <= next\_work\_state;

end

end

// calculate weight

always @(clo\_weight) begin

if (clo\_weight == 0) weight = 2;

else if (clo\_weight == 1) weight = 3;

else if (clo\_weight == 2) weight = 4;

else weight = 5;

end

always @(\*) begin

case (mode\_state)

0: begin //pre pausing mode

{power\_on, start} = 'b10;

hold = 1;

clear\_reg = 0;

ending = 0;

end\_reset = 0;

if (flag) begin

next\_mode\_state = 8;

next\_work\_state = 0;

end

else begin

flag\_reset = 0;

if (force\_end & reset) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end\_reset = 1;

end

else if (pause == 0) begin

next\_mode\_state = next\_mode;

next\_work\_state = work\_state;

end

else begin //pause == 1

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

end

end

1: begin //washing + rinsing + drying

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

{power\_on, start} = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b111;

stop = 0;

clear\_reg = 0;

ending = 0;

num1 = weight \* 4 + 21;

num3 = weight;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 1;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b111;

end\_reset = 1;

num2 = weight;

hold = 1;

next\_work\_state = 1;

recount = 'b11;

end

1: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b111;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 2;

num2 = 9;

recount = 'b10;

end

else begin

next\_work\_state = 1;

num2 = weight;

recount = 'b00;

end

end

2: begin //washing

{washing, rinsing, drying, water\_in, water\_out} = 'b10000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b111;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 3;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 2;

num2 = 9;

recount = 'b00;

end

end

3: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 4;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 3;

num2 = weight;

recount = 'b00;

end

end

4: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 5;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 4;

num2 = 3;

recount = 'b00;

end

end

5: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 6;

num2 = 6;

recount = 'b10;

end

else begin

next\_work\_state = 5;

num2 = weight;

recount = 'b00;

end

end

6: begin //rinsing

{washing, rinsing, drying, water\_in, water\_out} = 'b01000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 7;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 6;

num2 = 6;

recount = 'b00;

end

end

7: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 8;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 7;

num2 = weight;

recount = 'b00;

end

end

8: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 9;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 8;

num2 = 3;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

start = 0;

num2 = weight;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 9;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

2: begin //washing

{power\_on, start} = 'b11;

stop = 0;

clear\_reg = 0;

ending = 0;

num1 = weight + 9;

num3 = weight;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 2;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

end\_reset = 1;

num2 = weight;

hold = 1;

next\_work\_state = 1;

recount = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b100;

end

1: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b100;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 2;

num2 = 9;

recount = 'b10;

end

else begin

next\_work\_state = 1;

num2 = weight;

recount = 'b00;

end

end

2: begin //washing

{washing, rinsing, drying, water\_in, water\_out} = 'b10000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b100;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 3;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 2;

num2 = 9;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

num2 = weight;

start = 0;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 3;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

3: begin //washing + rinsing

stop = 0;

clear\_reg = 0;

ending = 0;

{power\_on, start} = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b110;

num1 = weight \* 3 + 18;

num3 = weight;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 3;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b110;

end\_reset = 1;

num2 = weight;

hold = 1;

next\_work\_state = 1;

recount = 'b11;

end

1: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b110;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 2;

num2 = 9;

recount = 'b10;

end

else begin

next\_work\_state = 1;

num2 = weight;

recount = 'b00;

end

end

2: begin //washing

{washing, rinsing, drying, water\_in, water\_out} = 'b10000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b110;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 3;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 2;

num2 = 9;

recount = 'b00;

end

end

3: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 4;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 3;

num2 = weight;

recount = 'b00;

end

end

4: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 5;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 4;

num2 = 3;

recount = 'b00;

end

end

5: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 6;

num2 = 6;

recount = 'b10;

end

else begin

next\_work\_state = 5;

num2 = weight;

recount = 'b00;

end

end

6: begin //rinsing

{washing, rinsing, drying, water\_in, water\_out} = 'b01000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 7;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 6;

num2 = 6;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

start = 0;

num2 = weight;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 7;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

4: begin //rinsing

{power\_on, start} = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

stop = 0;

clear\_reg = 0;

num1 = weight \* 2 + 9;

num3 = weight;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 4;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

num2 = weight;

end\_reset = 1;

hold = 1;

next\_work\_state = 3;

recount = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

end

3: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 4;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 3;

num2 = weight;

recount = 'b00;

end

end

4: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 5;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 4;

num2 = 3;

recount = 'b00;

end

end

5: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 6;

num2 = 6;

recount = 'b10;

end

else begin

next\_work\_state = 5;

num2 = weight;

recount = 'b00;

end

end

6: begin //rinsing

{washing, rinsing, drying, water\_in, water\_out} = 'b01000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 7;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 6;

num2 = 6;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

start = 0;

num2 = weight;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 7;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

5: begin //rinsing + drying

stop = 0;

clear\_reg = 0;

{power\_on, start} = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

num1 = weight \* 3 + 12;

num3 = weight;

ending = 0;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 5;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

num2 = weight;

end\_reset = 1;

hold = 1;

next\_work\_state = 3;

recount = 'b11;

end

3: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 4;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 3;

num2 = weight;

recount = 'b00;

end

end

4: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 5;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 4;

num2 = 3;

recount = 'b00;

end

end

5: begin //water in

{washing, rinsing, drying, water\_in, water\_out} = 'b00010;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 6;

num2 = 6;

recount = 'b10;

end

else begin

next\_work\_state = 5;

num2 = weight;

recount = 'b00;

end

end

6: begin //rinsing

{washing, rinsing, drying, water\_in, water\_out} = 'b01000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 7;

num2 = weight;

recount = 'b10;

end

else begin

next\_work\_state = 6;

num2 = 6;

recount = 'b00;

end

end

7: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

hold = 0;

end\_reset = 0;

if (time\_up[1]) begin

next\_work\_state = 8;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 7;

num2 = weight;

recount = 'b00;

end

end

8: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 9;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 8;

num2 = 3;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

start = 0;

num2 = weight;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 9;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

6: begin //drying

stop = 0;

{power\_on, start} = 'b11;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

ending = 0;

clear\_reg = 0;

num1 = weight + 3;

num3 = weight;

if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (pause == 1) begin

next\_mode\_state = 0;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 6;

case (work\_state)

0: begin //start

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

end\_reset = 1;

num2 = weight;

hold = 1;

next\_work\_state = 7;

recount = 'b11;

end

7: begin //water out

{washing, rinsing, drying, water\_in, water\_out} = 'b00001;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

end\_reset = 0;

hold = 0;

if (time\_up[1]) begin

next\_work\_state = 8;

num2 = 3;

recount = 'b10;

end

else begin

next\_work\_state = 7;

num2 = weight;

recount = 'b00;

end

end

8: begin //drying

{washing, rinsing, drying, water\_in, water\_out} = 'b00100;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

hold = 0;

end\_reset = 0;

if (time\_up[1] | time\_up[0]) begin

next\_work\_state = 9;

num2 = 0;

recount = 'b10;

end

else begin

next\_work\_state = 8;

num2 = 3;

recount = 'b00;

end

end

default: begin

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

start = 0;

num2 = weight;

recount = 'b00;

hold = 0;

ending = 1;

next\_work\_state = 9;

if (force\_end) begin

next\_mode\_state = 7;

end

else begin

next\_mode\_state = mode\_state;

end

end

endcase

end

end

7: begin //power off

{power\_on, start} = 'b00;

{washing, rinsing, drying, water\_in, water\_out} = 'b00000;

clear\_reg = 1;

mode\_wash = 0;

mode\_rinse = 0;

mode\_dry = 0;

end\_reset = 0;

hold = 0;

num1 = 0;

num2 = 0;

num3 = 0;

recount = 'b11;

if (force\_end & reset) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (reset == 1) begin

next\_mode\_state = 8;

next\_work\_state = 0;

end

else begin

next\_mode\_state = 7;

next\_work\_state = 0;

end\_reset = 1;

end

end

8: begin //pausing mode

{power\_on, start} = 'b10;

flag\_reset = 1;

clear\_reg = 0;

case (next\_mode)

0: begin

num1 = weight \* 4 + 21;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

end

1: begin

num1 = weight \* 4 + 21;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b111;

end

2: begin

num1 = weight + 9;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b100;

end

3: begin

num1 = weight \* 3 + 18;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b110;

end

4: begin

num1 = weight \* 2 + 9;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b010;

end

5: begin

num1 = weight \* 3 + 12;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b011;

end

6: begin

num1 = weight + 3;

num2 = weight;

num3 = weight;

{mode\_wash, mode\_rinse, mode\_dry} = 'b001;

end

default: begin

num1 = 0;

num2 = 0;

num3 = 0;

{mode\_wash, mode\_rinse, mode\_dry} = 'b000;

end

endcase

recount = 'b11;

hold = 1;

if (force\_end & reset) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end

else if (reset == 0 && child == 0) begin

next\_mode\_state = 7;

next\_work\_state = 0;

end\_reset = 1;

end

else if (pause == 0) begin

next\_mode\_state = next\_mode;

next\_work\_state = work\_state;

end

else begin

next\_mode\_state = 8;

next\_work\_state = work\_state;

end

end

endcase

end

endmodule

////////////////////////////////////////////////////// sel\_mode.v ////////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/12 08:05:51

// Design Name:

// Module Name: sel\_mode

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module sel\_mode(

input btn,

output reg [3:0] next\_mode

);

initial begin

next\_mode = 1;

end

always @(posedge btn) begin

if (next\_mode == 6) begin

next\_mode <= 1;

end

else begin

next\_mode <= next\_mode + 1;

end

end

endmodule

//////////////////////////////////////////////////////////////// countdown.v /////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/09 08:56:39

// Design Name:

// Module Name: countdown

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module countdown(

input clk\_k, // original 100MHz

input clk, // 1Hz clock

input [1:0] reset, // reset[0] : first number , reset[1] : second number

input hold\_wire, // 1~3 : 3 numbers hold

input [7:0] num1,

input [7:0] num2,

input [7:0] num3, // constant

output reg [1:0] final\_time\_up,

output [6:0] show\_num,

output [7:0] show\_port,

input clear

//debug

,output w1,

output w2,

output w3

);

reg [7:0] next1, next2;

wire [7:0] w1, w2, w3;

reg [1:0] zero;

reg [1:0] pause;

reg [1:0] time\_up;

always @(\*) final\_time\_up = time\_up & pause;

assign w3 = num3;

format\_disp fd (clk\_k, w1, w2, w3, show\_num, show\_port, clear);

assign w1 = next1;

assign w2 = next2;

initial begin

time\_up <= 0;

end

// if not reset, count the number

always @(posedge clk) begin

if (reset == 'b01) begin

next1 = num1;

end

else if (reset == 'b10) begin

next2 = num2;

end

else if (reset == 'b11) begin

next1 = num1;

next2 = num2;

end

else if (!hold\_wire) begin

if (next1 == 0) begin end

else if (next2 == 0) begin end

else begin

next1 <= next1 - 1;

next2 <= next2 - 1;

end

end

end

// calculate final\_time\_up

always @(negedge clk) begin

if ((!hold\_wire) && reset == 'b00 && (next1 == 0 || next2 == 0))

pause <= 'b11;

else pause <= 'b00;

end

// get time\_up when countdown finished

always @(posedge clk) begin

if ((!hold\_wire) && reset == 'b00 && (next1 == 0 || next2 == 0)) begin

if (next1 == 0) begin

time\_up[0] <= 1;

end

else if (next2 == 0) begin

time\_up[1] <= 1;

end

else time\_up = 'b00;

end

else begin

time\_up = 'b00;

end

end

endmodule

////////////////////////////////////////////////////////// divider.v //////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/05/18 14:15:35

// Design Name:

// Module Name: divider

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module divider(clk, clk\_N);

input clk;

output reg clk\_N;

parameter dely500us = 50\_000\_000;

//debug

//parameter dely500us = 8;

reg [31:0] counter;

initial begin

clk\_N = 0;

counter = 0;

end

always @(posedge clk) begin

if(counter==dely500us) begin

clk\_N <= ~clk\_N;

counter <= 0;

end

else begin

counter <= counter + 1;

end

end

endmodule

///////////////////////////////////////////////////////////// format\_disp ////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/09 01:20:00

// Design Name:

// Module Name: format\_disp

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module format\_disp(

input clk, // original clock 100MHz

input [7:0] num1,

input [7:0] num2,

input [7:0] num3,

output [6:0] show\_num,

output [7:0] show\_port,

input clear

);

wire [3:0] n1, n2, n3, n4, n5, n6, n7, n8;

assign n1 = num1 / 10;

assign n2 = num1 % 10;

assign n3 = 11; //invalid

assign n4 = num2 / 10;

assign n5 = num2 % 10;

assign n6 = 11; //invalid

assign n7 = num3 / 10;

assign n8 = num3 % 10;

disp\_num dn (clk, n1, n2, n3, n4, n5, n6, n7, n8, show\_num, show\_port, clear);

endmodule

/////////////////////////////////////////////////////////////// disp\_num /////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/09 01:05:42

// Design Name:

// Module Name: disp\_num

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module disp\_num(

input clk,

input [3:0] num1,

input [3:0] num2,

input [3:0] num3,

input [3:0] num4,

input [3:0] num5,

input [3:0] num6,

input [3:0] num7,

input [3:0] num8,

output [6:0] show\_num,

output [7:0] show\_port,

input clear

);

wire [6:0] n1, n2, n3, n4, n5, n6, n7, n8;

num2seg ns1 (num1, n1);

num2seg ns2 (num2, n2);

num2seg ns3 (num3, n3);

num2seg ns4 (num4, n4);

num2seg ns5 (num5, n5);

num2seg ns6 (num6, n6);

num2seg ns7 (num7, n7);

num2seg ns8 (num8, n8);

show\_seg ss (clk, n1, n2, n3, n4, n5, n6, n7, n8, show\_num, show\_port, clear);

endmodule

//////////////////////////////////////////////////////////////////// num2seg /////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/09 00:58:31

// Design Name:

// Module Name: num2seg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module num2seg(

input [3:0] num,

output reg [6:0] seg

);

always @(num) begin

case (num)

0: seg <= ~'b1111110;

1: seg <= ~'b0110000;

2: seg <= ~'b1101101;

3: seg <= ~'b1111001;

4: seg <= ~'b0110011;

5: seg <= ~'b1011011;

6: seg <= ~'b1011111;

7: seg <= ~'b1110000;

8: seg <= ~'b1111111;

9: seg <= ~'b1111011;

default: seg <= ~'b0000001;

endcase

end

endmodule

/////////////////////////////////////////////////////////////// show\_seg ////////////////////////////////////////////////////////////

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 2017/06/07 16:35:30

// Design Name:

// Module Name: show\_seg

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module show\_seg(

input clk,

input [6:0] num1,

input [6:0] num2,

input [6:0] num3,

input [6:0] num4,

input [6:0] num5,

input [6:0] num6,

input [6:0] num7,

input [6:0] num8,

output reg [6:0] final\_num,

output reg [7:0] show\_port,

input clear

);

reg [31:0] counter;

initial begin

counter = 0;

end

always @(posedge clk) begin

if (counter == 800000) counter <= 0;

else counter <= counter + 1;

end

always @(\*) begin

if (clear) begin

final\_num = num1;

show\_port = 'b11111111;

end

else if (counter < 100000) begin

final\_num = num1;

show\_port = 'b11111110;

end

else if (counter > 100000 && counter < 200000) begin

final\_num = num2;

show\_port = 'b11111101;

end

else if (counter > 200000 && counter < 300000) begin

final\_num = num3;

show\_port = 'b11111011;

end

else if (counter > 300000 && counter < 400000) begin

final\_num = num4;

show\_port = 'b11110111;

end

else if (counter > 400000 && counter < 500000) begin

final\_num = num5;

show\_port = 'b11101111;

end

else if (counter > 500000 && counter < 600000) begin

final\_num = num6;

show\_port = 'b11011111;

end

else if (counter > 600000 && counter < 700000) begin

final\_num = num7;

show\_port = 'b10111111;

end

else begin

final\_num = num8;

show\_port = 'b01111111;

end

end

endmodule