**1实验目的**

这次实验是通过所学习的Verilog语言与数字电路的相关知识，利用ISE继承开发环境并使用BASYS2来进行稍复杂的逻辑电路设计。

**2 实验要求**

基础要求：设计一个简单的自动售饮料机的逻辑电路。它的投币口每次只能投入一枚五 角或一元的硬币。投入一元五角钱硬币后机器自动给出一杯饮料。投入两元（两 枚一元）硬币后，在给出饮料的同时找出一枚五角的硬币。

扩展要求：

1、 用数码管显示输入的金额，以及要找出的金额。

2、 多物价系统。具体要求如下： 两个按钮，表示 5 角硬币和 1 元硬币。 可无限投入硬币，数码管动态显示当前金额（带小数）。 3 个按钮，代表 3 种饮料：可口可乐 2 元，午后红茶 3.5 元，乌龙茶 3 元。 每种饮料初始存货各 5 罐。 当按下某种饮料按钮后，如果投入钱币金额足够，则减去相应的金额，并以 数码管显示应找的钱币数目；如果不够，显示饮料价格并闪动，持续 2 秒， 然后仍然显示当前金额；如果饮料数目不够，用数码管显示。当按下退币按钮后，显示应找的钱币数目。

其他拓展：

除了上述扩展外，我还设计了补充货物功能，可以将三种饮料的库存重置为5.

为了程序中方便变量命名，在实验设计中，三种饮料分别取名为cola(￥2)，tea(￥3),milk(￥3.5)。下文分别用可乐、绿茶、牛奶代指三者。

**3设计分析**

本次实验设计的自动售货机一共会受到七种信号的激励，分别是：一元硬币、五角硬币、购买可乐、购买绿茶、购买牛奶、补充库存、退币。但是BASYS2板上的微动开关只有四个。在我最初的设计中，使用了读取拨码开关信号上升沿的方式来判断信号，但可能是因为板子本身硬件的问题，在拨动拨码开关从0到1的时候BASYS2板可能会读取两个上升沿，从1到0的时候也会读取到上升沿，带来很多操作上的困难。因此，在本次实验设计中，我使用拨码开关作为七种信号，但仅当按动微动开关BTN3时，板子才会读取信号。这样既满足了多种信号的要求，又便于操作

本实验设计中使用了BASYS2数码管，在货物不足的情况下会显示FFF.F，在投入硬币数不足以支付欲购买饮料的情况下会闪烁显示货物价格约2秒，退币时显示退币前瞬间的金额，其他情况下都是显示目前的金额。

此外，还采用了三个LED亮来分别表示三种饮料购买成功后的“出货”。通常情况下LED是熄灭的。

**4程序设计**

4.1顶层模块

module top\_module(

input clk,

input coin\_half,

input coin\_one,

input sell\_cola,

input sell\_tea,

input sell\_milk,

input full\_stock,

input account,

input coin\_return,

output reg cola\_out,

output reg tea\_out,

output reg milk\_out,

output[3:0]an,

output[6:0]a\_to\_g,

output DP

);

4.2LED模块

module led\_module(

input [3:0] NUM,

output reg [6:0] a\_to\_g

);

4.3管脚定义

表1 变量的管脚定义

|  |  |  |
| --- | --- | --- |
| **变 量** | **引 脚** | **对 应 功 能** |
| an[0]~an[3] | K14,F12,J12,M13 | 四个数码管显示 |
| account | A7 | 微动开关进行金额操作 |
| cola\_out,tea\_out, milk\_out | G1,P4,N4 | 相应的LED灯亮表示出货 |
| a\_to\_g[0]~[6] | L14、H12、N14、N11、P12、L13、M12 | 数码管的七段显示 |
| clk | B8 | 晶振输入时钟信号 |
| full\_stock | G3 | 补充货物 |
| coin\_return | B4 | 退币 |
| coin\_half, coin\_one | P11,L3 | 投入相应面值硬币 |
| sell\_cola,sell\_tea,sell\_milk | N3,E2,F3 | 购买相应饮料 |

**5实验结果**

在以下状况下分别按动account，结果如下：

1. sell\_\*(\*代表饮料名称)：若库存、现有硬币足够，扣除相应金额并电量相应LED。若库存不足，显示FFFF，硬币不足闪烁显示饮料金额两秒。
2. coin\_\*(\*代表面值)：数码管数字增加相应金额
3. return\_coin：数码管暂时无变化，下次操作按0元重新计数。
4. full\_stock:饮料库存被补充满

**6 心得体会**

这次实验使用的Verilog语言在以前没有正式地学习过，这次实验的也是在不断地学习、编译、报错、修改、调试中写出的。做完这次实验，也让我对Verilog这一门硬件描述语言有了更多的认识。

Verilog初看和C语言很相似，但作为硬件描述语言，Verilog和C语言又有很多细微的差别，这也导致了我最先编程的许多细节错误。在老师的实例程序与助教的帮助下，我逐渐学会了很多排除错误的方法，最终才得以完成程序的写作。

这次实验让我在实践中学习，有了不少平常课程带来不了的收获，我也要感谢老师与助教的悉心帮助！

**7 实验代码**

`timescale 1ns / 1ps

module led\_module(

input [3:0] NUM,

output reg [6:0] a\_to\_g

);

always @(\*)

case(NUM)

0:a\_to\_g=7'b0000001;

1:a\_to\_g=7'b1001111;

2:a\_to\_g=7'b0010010;

3:a\_to\_g=7'b0000110;

4:a\_to\_g=7'b1001100;

5:a\_to\_g=7'b0100100;

6:a\_to\_g=7'b0100000;

7:a\_to\_g=7'b0001111;

8:a\_to\_g=7'b0000000;

9:a\_to\_g=7'b0000100;

'hA: a\_to\_g=7'b0001000;

'hB: a\_to\_g=7'b1100000;

'hC: a\_to\_g=7'b0110001;

'hD: a\_to\_g=7'b1000010;

'hE: a\_to\_g=7'b0110000;

'hF: a\_to\_g=7'b0111000;

default: a\_to\_g=7'b0000001;

endcase

endmodule

module top\_module(

input clk,

input coin\_half,

input coin\_one,

input sell\_cola,

input sell\_tea,

input sell\_milk,

input full\_stock,

input account,

input coin\_return,

output reg cola\_out,

output reg tea\_out,

output reg milk\_out,

output[3:0]an,

output[6:0]a\_to\_g,

output DP

);

reg [32:0]clk\_cnt; //clock count

reg [3:0]NUM; //number printed on the segment LEDs

wire [1:0]s ;

wire q;

reg [2:0]stock\_cola;

reg [2:0]stock\_milk;

reg [2:0]stock\_tea;

reg [15:0]total\_money;

reg [15:0]display;

reg carry0;

reg carry1;

reg carry2;

reg carry3;

reg carry4;

reg timer;

reg [26:0]timer\_cnt;

reg not\_spark;

initial

begin

stock\_cola = 5;

stock\_milk = 5;

stock\_tea = 5;

carry0 = 0;

carry1 = 0;

carry2 = 0;

carry3 = 0;

carry4 = 0;

not\_spark = 0;

total\_money = 0;

clk\_cnt = 0;

timer = 0;

timer\_cnt = 0;

display = 0;

end

always @(posedge clk ) //deal the clock and clear events

begin

if(timer==0)

timer\_cnt = 0;

if (timer==1 && account == 1)

begin

timer\_cnt = 0;

timer\_cnt = timer\_cnt - 1;

end

clk\_cnt = clk\_cnt +1; //if clock flip, count clock

if(clk\_cnt[32:29]>15) //if count was full, back to zero

clk\_cnt = 0;

if(timer\_cnt != 0)

timer\_cnt = timer\_cnt - 1;

end

always @(posedge account)

begin

timer = 0;

if(coin\_one)

total\_money[7:4] = total\_money[7:4] + 1;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 10;

total\_money[7:4] = total\_money[7:4] + 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 10;

total\_money[11:8] = total\_money[11:8] + 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 10;

total\_money[15:12] = total\_money[15:12] + 1;

end

display = total\_money;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

if(coin\_half)

total\_money[3:0] = total\_money[3:0] + 5;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 10;

total\_money[7:4] = total\_money[7:4] + 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 10;

total\_money[11:8] = total\_money[11:8] + 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 10;

total\_money[15:12] = total\_money[15:12] + 1;

end

display = total\_money;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

if(full\_stock)

begin

stock\_cola = 5;

stock\_tea = 5;

stock\_milk = 5;

display = total\_money;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

if(coin\_return)

begin

display = total\_money;

total\_money = 0;

not\_spark = 0;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

if(sell\_cola)

begin

if(stock\_cola == 0)

begin

display = 'hFFFF;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else if(total\_money < 'h0020)

begin

display = 'h0020;

not\_spark = 0;

timer = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else

begin

total\_money[7:4] = total\_money[7:4] - 2;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 6;

total\_money[7:4] = total\_money[7:4] - 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 6;

total\_money[11:8] = total\_money[11:8] - 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 6;

total\_money[15:12] = total\_money[15:12] - 1;

end

if(total\_money[15:12]>9)

total\_money[15:12] = total\_money[15:12] - 6;

stock\_cola = stock\_cola - 1;

display = total\_money;

not\_spark = 1;

cola\_out = 1;

end

end

if(sell\_tea)

begin

if(stock\_tea == 0)

begin

display = 'hFFFF;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else if(total\_money < 'h0030)

begin

display = 'h0030;

not\_spark = 0;

timer = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else

begin

total\_money[7:4] = total\_money[7:4] - 3;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 6;

total\_money[7:4] = total\_money[7:4] - 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 6;

total\_money[11:8] = total\_money[11:8] - 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 6;

total\_money[15:12] = total\_money[15:12] - 1;

end

if(total\_money[15:12]>9)

total\_money[15:12] = total\_money[15:12] - 6;

stock\_tea = stock\_tea - 1;

display = total\_money;

not\_spark = 1;

tea\_out = 1;

end

end

if(sell\_milk)

begin

if(stock\_milk == 0)

begin

display = 'hFFFF;

not\_spark = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else if(total\_money < 'h0035)

begin

display = 'h0035;

not\_spark = 0;

timer = 1;

cola\_out = 0;

tea\_out = 0;

milk\_out = 0;

end

else

begin

total\_money[3:0] = total\_money[3:0] - 5;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 6;

total\_money[7:4] = total\_money[7:4] - 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 6;

total\_money[11:8] = total\_money[11:8] - 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 6;

total\_money[15:12] = total\_money[15:12] - 1;

end

if(total\_money[15:12]>9)

total\_money[15:12] = total\_money[15:12] - 6;

total\_money[7:4] = total\_money[7:4] - 3;

if(total\_money[3:0]>9)

begin

total\_money[3:0] = total\_money[3:0] - 6;

total\_money[7:4] = total\_money[7:4] - 1;

end

if(total\_money[7:4]>9)

begin

total\_money[7:4] = total\_money[7:4] - 6;

total\_money[11:8] = total\_money[11:8] - 1;

end

if(total\_money[11:8]>9)

begin

total\_money[11:8] = total\_money[11:8] - 6;

total\_money[15:12] = total\_money[15:12] - 1;

end

if(total\_money[15:12]>9)

total\_money[15:12] = total\_money[15:12] - 6;

stock\_milk = stock\_milk - 1;

display = total\_money;

not\_spark = 1;

milk\_out = 1;

end

end

end

assign s[1:0] = clk\_cnt[16:15]; //½»ÌæÏÔÊ¾

assign q = clk\_cnt[25];

assign an[0] = (s[0])||(s[1]) || (~not\_spark && q && timer\_cnt) ;

assign an[1] = (~s[0])||s[1] || (~not\_spark && q && timer\_cnt) ;

assign an[2] = s[0]||(~s[1]) || (~not\_spark && q && timer\_cnt) ;

assign an[3] = (~s[0])||(~s[1])|| (~not\_spark && q && timer\_cnt) ;

assign DP = (~s[0])||s[1] || (~not\_spark && q && timer\_cnt) ;

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

always @(\*)

case(s[1:0])

2'b00: NUM = display[3:0];

2'b01: NUM = display[7:4];

2'b10: NUM = display[11:8];

2'b11: NUM = display[15:12];

endcase

led\_module A1(.NUM(NUM),

.a\_to\_g(a\_to\_g));

endmodule