# 华中科技大学

# 《电子线路设计、测试与实验》实验报告

|  |  |
| --- | --- |
| 实验名称： | 数字系统设计 |
| 院（系）： | 电子信息与通信工程学院 |
| 专业班级： | 提高1701 |
| 姓名： | 周瑞松 |
| 学号： | U201713303 |
| 时间： | 2019.3.20 |
| 地点： | 南一楼东303 |
| 实验成绩： |  |
| 指导教师： | 汪小燕教授 |

2019 年 3月 20 日

实验任务及要求

基础要求：步进电机脉冲分配器（P241 设计任务7.3.2实验任务1）

提高要求：自动售货机

；

## 实验条件

|  |  |
| --- | --- |
| 内容 | 型号 |
| 开发环境 | ISE 14.7 |
| 开发语言 | Verilog HDL |
| 开发板 | NEXYS4 ARTIX-7 |

## 一、电路的设计过程

### 一、基础要求

步进电机脉冲分配器，可以分别正向反向输出。

#### 代码实现：

module dianji(

input reset,

output A,

output B,

output C,

input clk,

input M

);

reg A,B,C;

parameter st0=3'b001,st1=3'b011,st2=3'b010,st3=3'b110,st4=3'b100,st5=3'b101;

reg [2:0] pst,nst;

reg [25:0] fenpin = 0;

reg clk\_1ms = 0;

always @(posedge clk)

begin

if(fenpin == 100000000)

fenpin <= 0;

else

fenpin <= fenpin + 1'b1;

end

always @ (posedge clk)

begin

if(fenpin < 50000000)

clk\_1ms <= 0;

else

clk\_1ms <= 1;

end

always @ (posedge clk\_1ms)

begin

if(reset)

pst<=st0;

else

pst<=nst;

end

always @ (pst or M)

begin

if(!M)

begin

case(pst)

st0:nst<=st1;

st1:nst<=st2;

st2:nst<=st3;

st3:nst<=st4;

st4:nst<=st5;

st5:nst<=st0;

endcase

end

else

begin

case(pst)

st0:nst<=st5;

st1:nst<=st0;

st2:nst<=st1;

st3:nst<=st2;

st4:nst<=st3;

st5:nst<=st4;

endcase

end

end

always @(pst)

begin

case(pst)

st0:{C,B,A}<=st0;

st1:{C,B,A}<=st1;

st2:{C,B,A}<=st2;

st3:{C,B,A}<=st3;

st4:{C,B,A}<=st4;

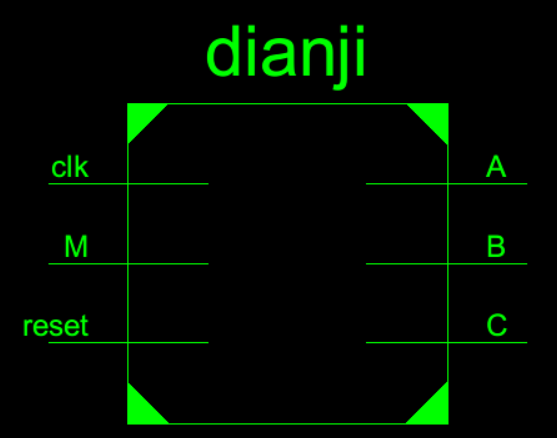
st5:{C,B,A}<=st5;

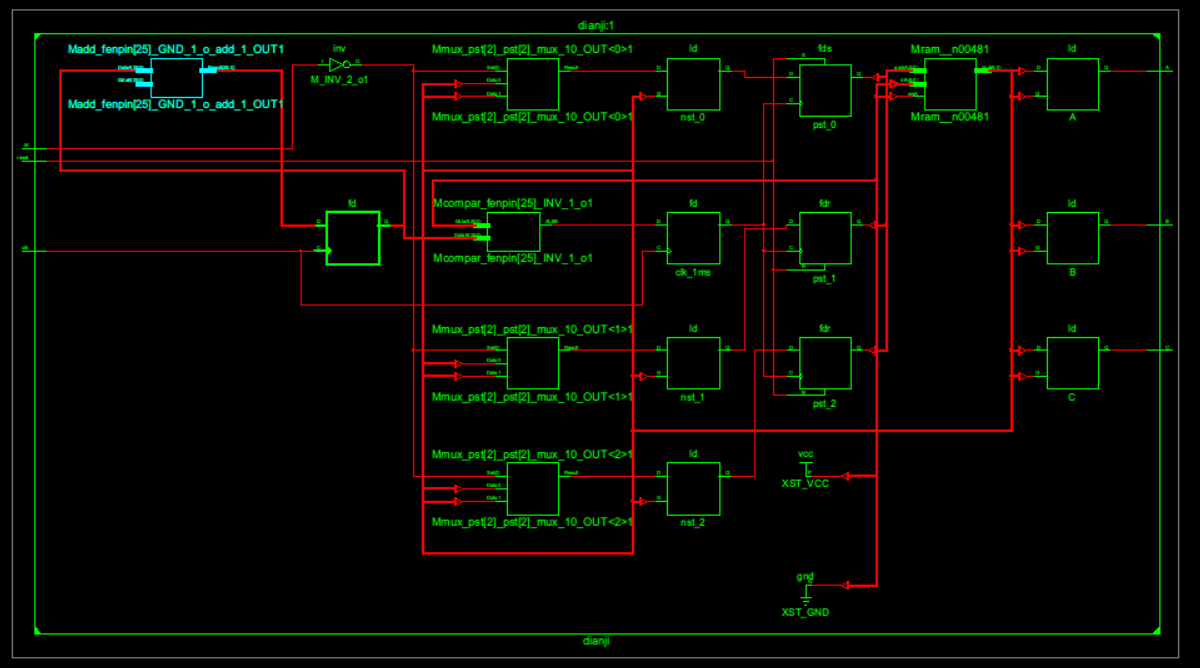
endcase

end

endmodule

#### 电路结构图：





### 二、提高要求

自动售卖机设计：

设计内容与要求

① 设计一自动售货机。预先设置4种商品，每种商品的价格不同。

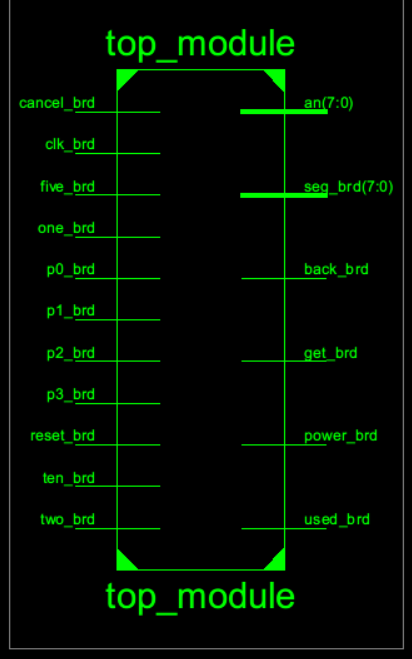
② 设计一模拟投币的方式，可以投币的种类有1元，2元，5元，10元。

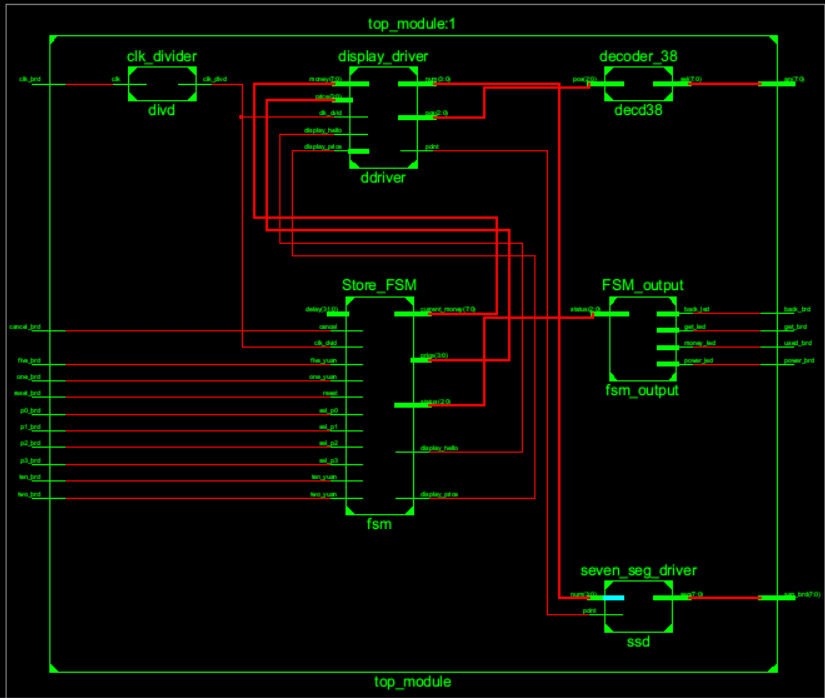
③ 设计一模拟找零的方式，可以找零的币种有1元，2元，5元。

④ 用Verilog语言对系统描述，能够在任意组合方式下完成商品的售卖和找零。设计进行描述及仿真，并下载到实验板上调试成功，适当增加必要的显示。

#### 代码结构：

#### 电路结构：





#### 代码实现：

/\*

\* 顶层模块，用于将开发板上io与模块绑定及连接各底层模块

\* @input params:

\* clk\_brd: 开发板上的clk输入

\* reset\_brd: 开发板上的reset输入

\* cancel\_brd: 开发板上的cancel输入

\* one\_brd: 开发板上的投币1元输入

\* two\_bird: 开发板上的投币2元输入

\* five\_bird:开发板上的投币5元输入

\* ten\_brd: 开发板上的投币10元输入

\* p0\_brd: 开发板上的商品1选择输入

\* p1\_brd: 开发板上的商品2选择输入

\* p2\_bird:开发板上的商品3选择输入

\* p3\_bird:开发板上的商品4选择输入

\* @output params:

\* power\_brd: 开发板上的电源指示灯

\* used\_brd: 开发板上的占用指示灯

\* get\_brd: 开发板上的取饮料指示灯

\* back\_brd: 开发板上的退币指示灯

\* seg\_brd: 开发板上的数码管

\* an: 开发板上的数码管片选信号

\*/

module top\_module(

input clk\_brd,

input reset\_brd,

input cancel\_brd,

input ten\_brd,

input one\_brd,

input five\_brd,

input two\_brd,

input p0\_brd,

input p1\_brd,

input p2\_brd,

input p3\_brd,

output power\_brd,

output used\_brd,

output get\_brd,

output back\_brd,

output [7:0]seg\_brd,

output [7:0]an

);

wire clk\_dvid;

wire [2:0]status; // fsm status

wire [7:0]current\_money; // current money

wire [3:0]price; // product price: 0 - no product, 1 - ￥1 product, 2 - ￥2 product 3 - ￥3 product 5 - ￥5 product

wire display\_hello; // display hello bit

wire display\_price; // display price bit

/\* 7-seg display position \*/

wire [2:0]pos;

wire [3:0]num;

wire point; // should display point

/\* clock divider module \*/

clk\_divider divd(clk\_brd, clk\_dvid);

/\* store fsm module \*/

Store\_FSM fsm(

.clk\_dvid(clk\_dvid),

.reset(reset\_brd),

.cancel(cancel\_brd),

.ten\_yuan(ten\_brd),

.one\_yuan(one\_brd),

.five\_yuan(five\_brd),

.two\_yuan(two\_brd),

.sel\_p0(p0\_brd),

.sel\_p1(p1\_brd),

.sel\_p2(p2\_brd),

.sel\_p3(p3\_brd),

.delay(3000),

/\* output \*/

.status(status),

.current\_money(current\_money),

.price(price),

.display\_hello(display\_hello),

.display\_price(display\_price)

);

/\* fsm output module, control leds \*/

FSM\_output fsm\_output(

.status(status),

.power\_led(power\_brd),

.money\_led(used\_brd),

.get\_led(get\_brd),

.back\_led(back\_brd)

);

/\* display driver, output 7-segment led \*/

display\_driver ddriver(

.clk\_dvid(clk\_dvid),

.price(price),

.money(current\_money),

.display\_hello(display\_hello),

.display\_price(display\_price),

.pos(pos),

.num(num),

.point(point)

);

/\* 3-8 decoder module, control an \*/

decoder\_38 decd38(.pos(pos), .sel(an));

/\* seven segment driver module \*/

seven\_seg\_driver ssd(.num(num), .point(point), .seg(seg\_brd));

endmodule

/\*

\* 时钟分频器模块， 分频比：1/delay

\* input params:

\* clk: 时钟输入

\* output params:

\* clk\_divd: 分频后的时钟信号

\*/

module clk\_divider(

input clk,

output reg clk\_divd

//output reg clk\_refresh

);

reg [31:0]count;

//reg [31:0]count2;

parameter delay = 100\_000;

initial begin

count = 0;

clk\_divd = 0;

// clk\_refresh =0;

end

always@ (posedge clk) begin

if (count >= delay / 2) begin

count = 0;

clk\_divd = ~clk\_divd;

end

count = count + 1;

end

endmodule

/\* status

\* 001 - power on init

\* 010 - money is not enough to buy

\* 011 - add money

\* 100 - display money

\* 101 - get product

\* 110 - pay back money

\*/

/\*

\* @input params:

\* clk\_dvid: clock signal

\* reset: reset the system

\* cancel: remove the money

\* ten\_yuan: drop 10 yuan

\* one\_yuan: drop 1 yuan

\* five\_yuan: drop 5 yuan

\* two\_yuan: drop 2 yuan

\* sel\_p0: sell goods 0

\* sel\_p1: sell goods 1

\* delay: 取货及退币的延时参数（ms）

\* @output params:

\* power\_led: 电源指示灯

\* used\_led: 占用指示灯

\* get\_led: 取饮料指示灯

\* back\_led: 退币指示灯

\* seg: 数码管信号

\* seg\_pos: 数码管片选信号

\*/

module Store\_FSM(

input clk\_dvid,

input reset,

input cancel,

input ten\_yuan,

input one\_yuan,

input two\_yuan,

input five\_yuan,

input sel\_p0,

input sel\_p1,

input sel\_p2,

input sel\_p3,

input [31:0]delay,

output reg [2:0]status,

output reg [7:0]current\_money,

output reg [3:0]price,

output reg display\_hello,

output reg display\_price

);

// parameter delay = 2000;

reg [31:0]time\_count; // time count for status 5&6 delay

/\* key pressed flags \*/

reg cancel\_pressed, clear\_flag, one\_pressed,two\_pressed,five\_pressed, ten\_pressed;

reg sel0\_pressed, sel1\_pressed,sel2\_pressed,sel3\_pressed;

initial begin

status = 3'b000;

current\_money = 0;

price = 0;

time\_count = 0;

cancel\_pressed = 0;

sel0\_pressed = 0;

sel1\_pressed = 0;

sel2\_pressed = 0;

sel3\_pressed = 0;

clear\_flag = 0;

one\_pressed = 0;

two\_pressed = 0;

five\_pressed = 0;

ten\_pressed = 0;

display\_hello = 1;

display\_price = 0;

end

/\* one yuan input key \*/

always@ (posedge one\_yuan or posedge clear\_flag) begin

if (clear\_flag == 1) begin

one\_pressed = 0;

end

else begin

one\_pressed = 1;

end

end

/\* two yuan input key \*/

always@ (posedge two\_yuan or posedge clear\_flag) begin

if (clear\_flag == 1) begin

two\_pressed = 0;

end

else begin

two\_pressed = 1;

end

end

/\* five yuan input key \*/

always@ (posedge five\_yuan or posedge clear\_flag) begin

if (clear\_flag == 1) begin

five\_pressed = 0;

end

else begin

five\_pressed = 1;

end

end

/\* ten yuan input key \*/

always@ (posedge ten\_yuan or posedge clear\_flag) begin

if (clear\_flag == 1) begin

ten\_pressed = 0;

end

else begin

ten\_pressed = 1;

end

end

/\* cancel input key \*/

always@ (posedge cancel or posedge clear\_flag) begin

if (clear\_flag == 1) begin

cancel\_pressed = 0;

end

else if (status != 3'b001) begin

cancel\_pressed = 1;

end

end

/\* select product 1 input key \*/

always@ (posedge sel\_p0 or posedge clear\_flag) begin

if (clear\_flag == 1) begin

sel0\_pressed = 0;

end

else begin

sel0\_pressed = 1;

end

end

/\* select product 2 input key \*/

always@ (posedge sel\_p1 or posedge clear\_flag) begin

if (clear\_flag == 1) begin

sel1\_pressed = 0;

end

else begin

sel1\_pressed = 1;

end

end

/\* select product 3 input key \*/

always@ (posedge sel\_p2 or posedge clear\_flag) begin

if (clear\_flag == 1) begin

sel2\_pressed = 0;

end

else begin

sel2\_pressed = 1;

end

end

/\* select product 4 input key \*/

always@ (posedge sel\_p3 or posedge clear\_flag) begin

if (clear\_flag == 1) begin

sel3\_pressed = 0;

end

else begin

sel3\_pressed = 1;

end

end

/\* main \*/

always@ (posedge clk\_dvid) begin

if (reset) begin

/\* if reset on while startup \*/

status = 3'b001;

display\_hello = 1;

current\_money = 0;

end

if (status == 3'b001 || status == 3'b000) begin

/\* initial state -- display hello \*/

display\_hello = 1;

end

else begin

display\_hello = 0;

end

if (clear\_flag == 1) begin

clear\_flag = 0;

end

/\* cancel key down \*/

if (cancel\_pressed == 1) begin

status = 3'b110;

time\_count = 0;

clear\_flag = 1;

end

/\* sel0 key down \*/

if (sel0\_pressed == 1) begin

if(current\_money > 1)

begin

clear\_flag = 1;

price = 1;

time\_count = 0;

status = 3'b101;

current\_money = current\_money - 1;

end

else begin

clear\_flag = 1;

time\_count = 0;

status = 3'b010;

display\_price = 1;

end

end

/\* sel1 key down \*/

if (sel1\_pressed == 1) begin

if(current\_money > 2)

begin

clear\_flag = 1;

price = 2;

time\_count = 0;

status = 3'b101;

current\_money = current\_money - 2;

end

else begin

clear\_flag = 1;

time\_count = 0;

status = 3'b010;

display\_price = 1;

end

end

/\* sel2 key down \*/

if (sel2\_pressed == 1) begin

if(current\_money > 3)

begin

clear\_flag = 1;

price = 3;

time\_count = 0;

status = 3'b101;

current\_money = current\_money - 3;

end

else begin

clear\_flag = 1;

time\_count = 0;

status = 3'b010;

display\_price = 1;

end

end

/\* sel3 key down \*/

if (sel3\_pressed == 1) begin

if(current\_money > 5)

begin

clear\_flag = 1;

price = 5;

time\_count = 0;

status = 3'b101;

current\_money = current\_money - 5;

end

else begin

clear\_flag = 1;

time\_count = 0;

status = 3'b010;

display\_price = 1;

end

end

/\* one yuan input key down \*/

if (one\_pressed == 1) begin

clear\_flag = 1;

current\_money = current\_money + 1;

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

/\* two yuan input key down \*/

if (two\_pressed == 1) begin

clear\_flag = 1;

current\_money = current\_money + 2;

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

/\* five yuan input key down \*/

if (five\_pressed == 1) begin

clear\_flag = 1;

current\_money = current\_money + 5;

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

/\* ten yuan input key down \*/

if (ten\_pressed == 1) begin

clear\_flag = 1;

current\_money = current\_money + 10;

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

/\* delay time countup \*/

time\_count = time\_count + 1;

if (time\_count == delay) begin

if (status == 3'b101) begin

/\* get product done \*/

status = 3'b011;

time\_count = 0;

price = 0;

display\_price = 0;

end

else if (status == 3'b110) begin

/\* pay back money done \*/

status = 3'b001;

current\_money = 0;

price = 0;

display\_price = 0;

end

else if (status == 3'b010) begin

/\*money is not enough\*/

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

else if (status == 3'b011) begin

/\*display price\*/

status = 3'b100;

time\_count = 0;

display\_price = 0;

end

else if (status == 3'b100) begin

/\*display money\*/

status = 3'b011;

time\_count = 0;

display\_price = 1;

end

end

end

endmodule

/\*

\* input params:

\* status: 当前FSM状态

\* output params:

\* power\_led: 电源指示灯

\* money\_led: 占用指示灯

\* get\_led: 取饮料指示灯

\* back\_led: 退币指示灯

\*/

module FSM\_output(

input [2:0]status,

output reg power\_led,

output reg money\_led,

output reg get\_led,

output reg back\_led);

always@ (status) begin

case (status)

3'b000: begin

power\_led = 0;

money\_led = 0;

get\_led = 0;

back\_led = 0;

end

3'b001: begin

power\_led = 1;

money\_led = 0;

get\_led = 0;

back\_led = 0;

end

3'b010: begin

power\_led = 1;

money\_led = 1;

get\_led = 0;

back\_led = 0;

end

3'b011: begin

power\_led = 1;

money\_led = 1;

get\_led = 0;

back\_led = 0;

end

3'b100: begin

power\_led = 1;

money\_led = 1;

get\_led = 0;

back\_led = 0;

end

3'b101: begin

power\_led = 1;

money\_led = 0;

get\_led = 1;

back\_led = 0;

end

3'b110: begin

power\_led = 1;

money\_led = 0;

get\_led = 0;

back\_led = 1;

end

default: begin

power\_led = 0;

money\_led = 0;

get\_led = 0;

back\_led = 0;

end

endcase

end

endmodule

/\*

\* 数码管显示驱动模块， 用于将数值转化为数码管输出

\* input params:

\* clk\_dvid: 数码管动态显示时钟

\* price: 商品价格， 0 - no product,

1 - ￥1 product,

2 - ￥2 product

3 - ￥3 product

4 - ￥5 product

\* money: 当前金额

\* display\_hello: 是否显示"HELLO"

\* display\_price:

\* output params:

\* seg: 数码管显示信号

\* seg\_pos: 数码管片选信号

\*/

module display\_driver(

input clk\_dvid,

input [3:0]price,

input [7:0]money,

input display\_hello,

input display\_price,

//input close\_display,

output reg [2:0]pos,

output reg [3:0]num,

output reg point

);

/\* save display nums \*/

reg [3:0] p0, p1, p2, m0, m1, m2, p3, tmp;

initial begin

point = 0;

pos = 0;

num = 4'b0000;

p0 = 0;

p1 = 0;

p2 = 0;

m0 = 0;

m1 = 0;

m2 = 0;

p3 = 14;

tmp = 0;

end

always@ (posedge clk\_dvid)

begin

case (pos)

0: begin

point = 1;

num = m0;

end

1: begin

point = 0;

num = m1;

end

2: begin

point = 1;

num = m2;

end

3: begin

point = 0;

num = p0;

end

4: begin

point = 1;

num = p1;

end

5: begin

point = 0;

num = p2;

end

6: begin

num = 14;

end

7: begin

num = p3;

end

endcase

pos = pos + 1;

end

always@ (price or display\_hello or display\_price) begin

if (display\_hello == 1) begin

p2 = 13;

p1 = 12;

p0 = 12;

p3 = 14;

end

else if (display\_price == 1) begin

p3 = 1;

p0 = 3;

p2 = 5;

p1 = 14;

end

else begin

p0 = (money % 5) / 2;

p1 = 1;

p2 = money - 5\*m1 - 2\*p0;

p3 = 14;

end

end

always@ (money or display\_hello or display\_price) begin

if (display\_hello == 1) begin

m0 = 14;

m1 = 10;

m2 = 11;

end

else if (display\_price == 1) begin

m0 = 14;

m1 = 2;

m2 = 14;

end

else begin

m0 = 5;

m2 = 2;

m1 = money / 5;

end

end

endmodule

/\* 3-8译码器模块 \*/

module decoder\_38(

input [2:0]pos,

output reg [7:0]sel);

always@ (pos) begin

case (pos)

3'b000: sel = 8'b11111110;

3'b001: sel = 8'b11111101;

3'b010: sel = 8'b11111011;

3'b011: sel = 8'b11110111;

3'b100: sel = 8'b11101111;

3'b101: sel = 8'b11011111;

3'b110: sel = 8'b10111111;

3'b111: sel = 8'b01111111;

default: sel = 8'b11111111;

endcase

end

endmodule

/\*

\* <-- led map -->

\* center - 6

\* right top - 1

\* top - 0

\* right bottom - 2

\* left top - 5

\* left bottom - 4

\* bottom - 3

\*/

/\*

\* 七段数码管显示驱动模块

\* @input params:

\* num: 显示的数值, 0-9:数字0-9, 10:H, 11:E, 12:L, 13:O, 其它：不显示

\* point: 是否显示小数点

\* @output parans:

\* seg: 数码管显示信号

\*/

module seven\_seg\_driver(

input [3:0]num,

input point,

output reg [7:0]seg);

always@ (num or point) begin

case (num)

0: begin

if (point == 0)

seg = 8'b11000000;

else

seg = 8'b01000000;

end

1: begin

if (point == 0)

seg = 8'b11111001;

else

seg = 8'b01111001;

end

2: begin

if (point == 0)

seg = 8'b10100100;

else

seg = 8'b00100100;

end

3: begin

if (point == 0)

seg = 8'b10110000;

else

seg = 8'b00110000;

end

4: begin

if (point == 0)

seg = 8'b10011001;

else

seg = 8'b00011001;

end

5: begin

if (point == 0)

seg = 8'b10010010;

else

seg = 8'b00010010;

end

6: begin

if (point == 0)

seg = 8'b10000010;

else

seg = 8'b00000010;

end

7: begin

if (point == 0)

seg = 8'b11111000;

else

seg = 8'b01111000;

end

8: begin

if (point == 0)

seg = 8'b10000000;

else

seg = 8'b00000000;

end

9: begin

if (point == 0)

seg = 8'b10010000;

else

seg = 8'b00010000;

end

10: begin

/\* H \*/

seg = 8'b10001001;

end

11: begin

/\* E \*/

seg = 8'b10000110;

end

12: begin

/\* L \*/

seg = 8'b11000111;

end

13: begin

/\* O - same as number 0 \*/

seg = 8'b11000000;

end

default: begin

seg = 8'b11111111;

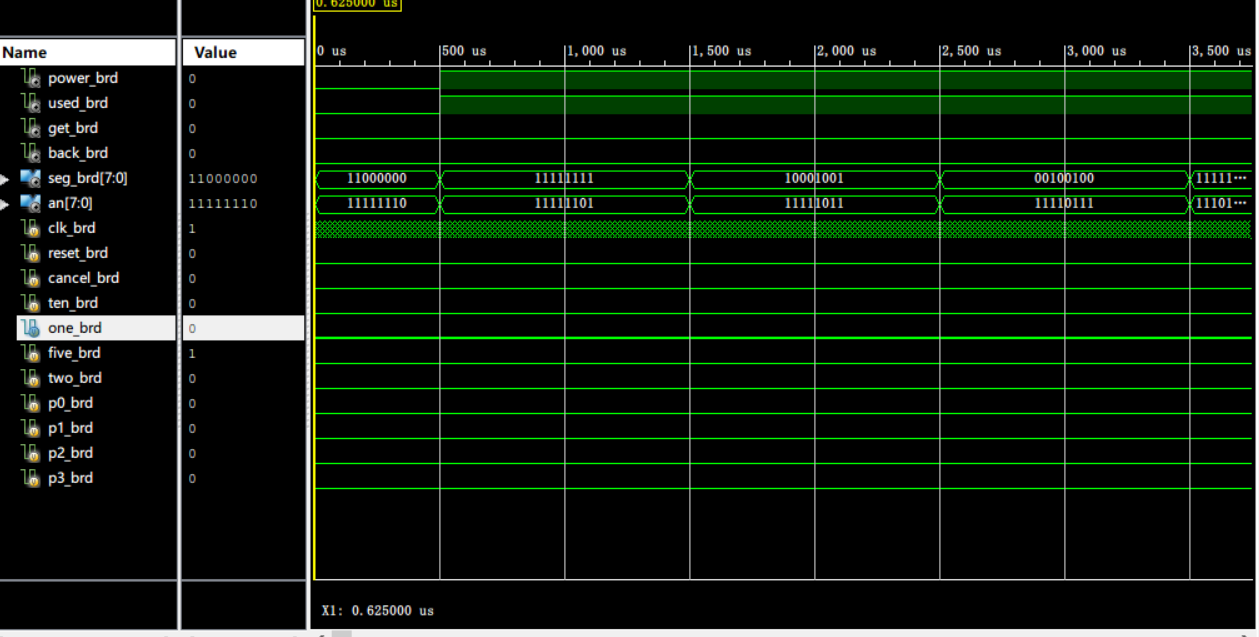
end

endcase

end

endmodule

#### 仿真结果：



#### 使用说明：

自动售卖机初始化时，数码显示管会显示HELLO，自动售卖机可以选择收取1,2,5,10元的钱，并内置了分别为1,2,3,5元的商品。在开发板上，分别用四位开关代表四种投币和购买四种商品的行为。并且各种行为伴有提示灯的改变。分别有电源指示灯，占用指示灯，取饮料指示灯，退币指示灯对应FPGA板上N14，J13，K15，H17位LED。数码显示管默认两秒切换一次输出，输出有两种，分别为现有的投币和商品价格。若钱币不足以购买商品，商品价格会立即输出，若有投币行为，钱币数会立即输出。购买完成后，按N17按键进行退币。

#### 资源占用：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slice Logic Utilization** | **Used** | **Available** | **Utilization** | **Note(s)** |
| Number of Slice Registers | 97 | 126,800 | 1% |  |
| Number used as Flip Flops | 97 |  |  |  |
| Number used as Latches | 0 |  |  |  |
| Number used as Latch-thrus | 0 |  |  |  |
| Number used as AND/OR logics | 0 |  |  |  |
| Number of Slice LUTs | 291 | 63,400 | 1% |  |
| Number used as logic | 291 | 63,400 | 1% |  |
| Number using O6 output only | 195 |  |  |  |
| Number using O5 output only | 0 |  |  |  |
| Number using O5 and O6 | 96 |  |  |  |
| Number used as ROM | 0 |  |  |  |
| Number used as Memory | 0 | 19,000 | 0% |  |
| Number used exclusively as route-thrus | 0 |  |  |  |
| Number of occupied Slices | 106 | 15,850 | 1% |  |
| Number of LUT Flip Flop pairs used | 299 |  |  |  |
| Number with an unused Flip Flop | 208 | 299 | 69% |  |
| Number with an unused LUT | 8 | 299 | 2% |  |
| Number of fully used LUT-FF pairs | 83 | 299 | 27% |  |
| Number of unique control sets | 16 |  |  |  |
| Number of slice register sites lost         to control set restrictions | 95 | 126,800 | 1% |  |
| Number of bonded [IOBs](F://FPGAcode/vendor_machine/top_module_map.xrpt?&DataKey=IOBProperties) | 31 | 210 | 14% |  |
| Number of LOCed IOBs | 31 | 31 | 100% |  |
| Number of RAMB36E1/FIFO36E1s | 0 | 135 | 0% |  |
| Number of RAMB18E1/FIFO18E1s | 0 | 270 | 0% |  |
| Number of BUFG/BUFGCTRLs | 11 | 32 | 34% |  |
| Number used as BUFGs | 11 |  |  |  |
| Number used as BUFGCTRLs | 0 |  |  |  |
| Number of IDELAYE2/IDELAYE2\_FINEDELAYs | 0 | 300 | 0% |  |
| Number of ILOGICE2/ILOGICE3/ISERDESE2s | 0 | 300 | 0% |  |
| Number of ODELAYE2/ODELAYE2\_FINEDELAYs | 0 |  |  |  |
| Number of OLOGICE2/OLOGICE3/OSERDESE2s | 0 | 300 | 0% |  |
| Number of PHASER\_IN/PHASER\_IN\_PHYs | 0 | 24 | 0% |  |
| Number of PHASER\_OUT/PHASER\_OUT\_PHYs | 0 | 24 | 0% |  |
| Number of BSCANs | 0 | 4 | 0% |  |
| Number of BUFHCEs | 0 | 96 | 0% |  |
| Number of BUFRs | 0 | 24 | 0% |  |
| Number of CAPTUREs | 0 | 1 | 0% |  |
| Number of DNA\_PORTs | 0 | 1 | 0% |  |
| Number of DSP48E1s | 0 | 240 | 0% |  |
| Number of EFUSE\_USRs | 0 | 1 | 0% |  |
| Number of FRAME\_ECCs | 0 | 1 | 0% |  |
| Number of IBUFDS\_GTE2s | 0 | 4 | 0% |  |
| Number of ICAPs | 0 | 2 | 0% |  |
| Number of IDELAYCTRLs | 0 | 6 | 0% |  |
| Number of IN\_FIFOs | 0 | 24 | 0% |  |
| Number of MMCME2\_ADVs | 0 | 6 | 0% |  |
| Number of OUT\_FIFOs | 0 | 24 | 0% |  |
| Number of PCIE\_2\_1s | 0 | 1 | 0% |  |
| Number of PHASER\_REFs | 0 | 6 | 0% |  |
| Number of PHY\_CONTROLs | 0 | 6 | 0% |  |
| Number of PLLE2\_ADVs | 0 | 6 | 0% |  |
| Number of STARTUPs | 0 | 1 | 0% |  |
| Number of XADCs | 0 | 1 | 0% |  |
| Average Fanout of Non-Clock Nets | 4.11 |  |  |  |

## 二、实验结果及分析

经过仿真检查，电路逻辑正确，因此编译生成bit文件写入开发板中，各项功能功能运行正常。资源利用率正常。

## 三、小结

本次试验综合性较强，一开始由于没有仔细去设计构想，导致代码越写越乱，把门级电路描述和行为描述弄混了，导致程序很乱，除了很多错误。之后重新设计了一番框架模型，将每个模块要实现的功能与输入输出明细，然后重新开始构建每个模块，最后实现了所有功能。可见，在电路设计中，一定要有目的性，要有很明细的架构设计。

本次试验采用了门级描述和行为级描述的结合，在顶层模块使用门级描述，在分模块中使用更为方便的行为级描述。

最重要的是，一写报告的规范的到了训练，从实验介绍，到功能说明以及仿真报告都需要认真地完成，不得马虎。