**西安电子科技大学**

**数字电子技术基础 课程设计报告**

**实验名称**

机电工程学院 学院 1504032 班

成 绩

姓名 吕昕远 学号 15040310079

同作者

实验日期 2017 年 10 月 9 日

|  |
| --- |
| 指导教师评语：  指导教师：  年 月 日 |
| **实验报告内容基本要求及参考格式**  一、实验目的  二、实验所用仪器（或实验环境）  三、实验基本原理及步骤（或方案设计及理论计算）  四、实验数据记录（或仿真及软件设计）  五、实验结果分析及回答问题（或测试环境及测试结果） |

一、实验目的

1. 掌握模数、数模转换相关概念

2. 实现FPGA对ADC和DAC的控制

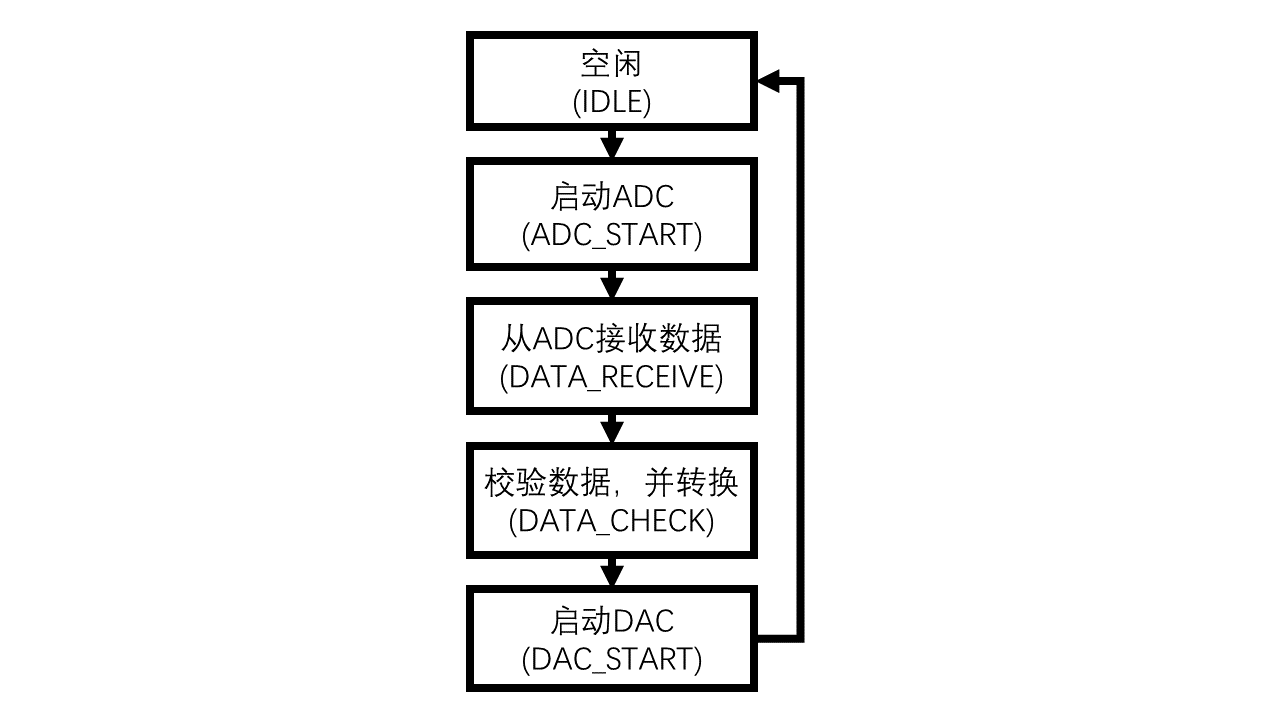
二、实验所用仪器（或实验环境）

Quartus II 9.1

三、实验基本原理及步骤（或方案设计及理论计算）

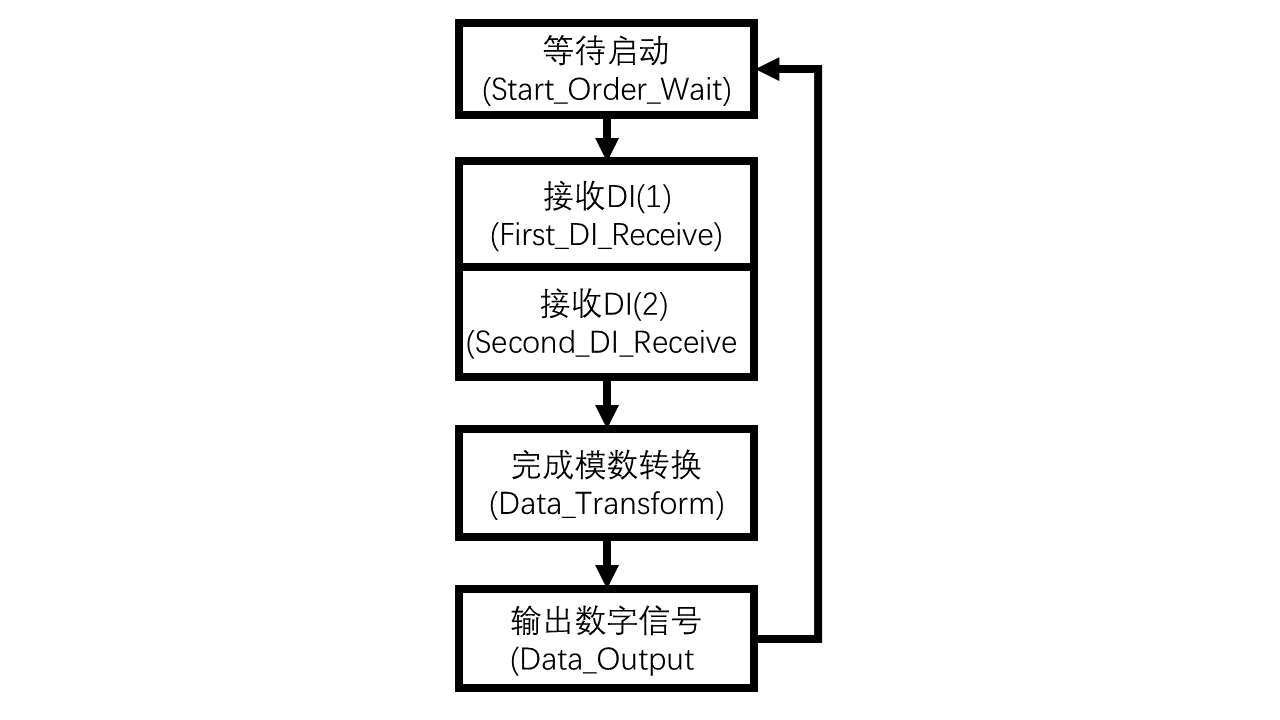
控制器部分采用有限状态机形式进行设计，并模拟ADC0832工作时序，**实现了“ADC0832”测试模块**。

控制器状态转移如下图：



为了更好的测试控制器的工作状态，参照ADC0832的工作时序[1]和使用方法[2]实现了“ADC0832”测试模块。

“ADC0832”测试模块工作状态转换如下图



四、实验数据记录（或仿真及软件设计）

**控制器（AD\_DA.vhd[3]）**

library ieee;

use ieee.STD\_LOGIC\_ARITH.all;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity AD\_DA is

port(

--控制ADC0832的相关端口

-- port of ADC0832

clk\_1 : in std\_logic; --clock

CH0\_1 : in integer; --analog data entrance

CH1\_1 : in integer; --analog data entrance

CS\_output : out std\_logic;

DI\_output : out std\_logic;

state\_signal\_1 : buffer std\_logic\_vector(3 downto 0); --状态编号

DO\_1 : buffer std\_logic; --Data Out

DataA\_output: out std\_logic\_vector(7 downto 0);

DataB\_output: out std\_logic\_vector(7 downto 0);

stu\_no:buffer integer; --学号

--控制DAC0832的相关端口

--port of DAC0832

D : out std\_logic\_vector(7 downto 0);

WR1 : out std\_logic;

XFER : out std\_logic;

WR2 : out std\_logic

);

signal DI\_1 : std\_logic:='0';

signal CS\_1 : std\_logic:='1'; --使能端

end AD\_DA;

architecture behavior of AD\_DA is

--ADC0832测试模块

COMPONENT ADC0832

port(

clk : in std\_logic; --clock

CH0,CH1 : in integer; --analog data entrance

CS : in std\_logic; --CHIP Select

DI : In std\_logic; --Data In

state\_signal : buffer std\_logic\_vector(3 downto 0); --state Out

DO : buffer std\_logic --Data Out

);

END COMPONENT ADC0832;

--状态定义

type state is (IDLE,ADC\_START,DATA\_RECEIVE,DATA\_CHECK,DAC\_START);

signal current\_state:state; --现态

signal next\_state:state; --次态

signal DataA : std\_logic\_vector(7 downto 0); --正向接收数据

signal DataB : std\_logic\_vector(7 downto 0); --反向接收数据

signal receive\_order:std\_logic; --接收数据进程启动的控制信号

signal start\_order:std\_logic; --ADC启动的控制信号

signal temp:integer;

signal Parallel\_Data:std\_logic\_vector(7 downto 0); --并行数据

shared VARIABLE COUNT1:INTEGER:=0;

shared VARIABLE COUNT2:INTEGER:=0;

shared VARIABLE COUNT3:INTEGER:=0;

begin

--实例化“ADC0832”

u1:ADC0832 PORT MAP(clk=>clk\_1, CH0=>CH0\_1, CH1=>CH1\_1, CS=>CS\_1,DI=>DI\_1,state\_signal=>state\_signal\_1,DO=>DO\_1);

--时序电路

counter : process(clk\_1)

begin

if(clk\_1'event and clk\_1 = '0') then

current\_state <= next\_state;

end if;

end process;

--组合电路

controller : process(current\_state)

begin

case current\_state is

when IDLE =>

--初始化各个信号

--init the signal

CS\_1 <= '0';

WR1 <= '1';

receive\_order <= '0';

start\_order <= '0';

--DI\_1 <= '0';

--设置次态

next\_state <= ADC\_START;

when ADC\_START =>

--启动ADC0832

--send start order to ADC0832

if(COUNT2 < 4) then

start\_order <= '1';

next\_state <= ADC\_START;

else

start\_order <= '0';

next\_state <= DATA\_RECEIVE;

end if;

when DATA\_RECEIVE =>

--从ADC0832接收转换完成的串行数据

--receive data from ADC0832

if(COUNT1 < 15) then

--启动数据接收进程

receive\_order <= '1';

next\_state <= DATA\_RECEIVE;

else

receive\_order <= '0';

next\_state <= DATA\_CHECK;

end if;

when DATA\_CHECK =>

--校验数据

--check if the data is correct

if(DataA = DataB) then

--计算出学号

temp <= CONV\_INTEGER(DataA);

stu\_no <= (255 - temp)/2;

next\_state <= DAC\_START;

else

stu\_no <= -1;

next\_state <= DAC\_START;

end if;

Parallel\_Data <= CONV\_STD\_LOGIC\_VECTOR((255 - temp)/2,8);

when DAC\_START =>

--启动DAC0832

--send start order to DAC0832

D <= Parallel\_Data;

WR1 <= '0';

WR2 <= '0';

XFER <= '0';

next\_state <= DAC\_START;

when others =>

next\_state <= IDLE;

end case;

DataA\_output <= DataA;

DataB\_output <= DataB;

CS\_output <= CS\_1;

DI\_output <= DI\_1;

end process;

--ADC0832控制进程

adc\_work : process(start\_order,clk\_1)

begin

if(start\_order = '1') then

if(clk\_1'event and clk\_1 = '0') then

if(COUNT2 = 0) then

--START\_BIT

DI\_1 <= '1';

end if;

if(COUNT2 = 1) then

--First DI

DI\_1 <= '0';

end if;

if(COUNT2 = 2) then

--Second DI

DI\_1 <= '1';

end if;

if(COUNT2 = 3) then

--Transform

DI\_1 <= '0';

end if;

COUNT2:=COUNT2+1;

end if;

end if;

end process;

--接收ADC0832 数据进程

receive : process(receive\_order,clk\_1)

begin

if(receive\_order = '1') then

if(clk\_1'event and clk\_1='0') then

--先从高位到低位，再从高位到低位

if(COUNT1<7) then

DataA <= DataA(6 downto 0) & DO\_1;

end if;

if(COUNT1=7) then

DataA <= DataA(6 downto 0) & DO\_1;

DataB <= DO\_1 & DataB(7 downto 1);

end if;

if(COUNT1<15 and COUNT1 >7) then

DataB <= DO\_1 & DataB(7 downto 1);

end if;

COUNT1:=COUNT1+1;

end if;

end if;

end process;

end behavior;

**ADC0832测试模块（ADC0832.vhd[4]）**

library ieee;

use ieee.STD\_LOGIC\_ARITH.all;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity ADC0832 is

port(

clk : in std\_logic;

CH0,CH1 : in integer; --“模拟”信号输入

CS : in std\_logic; --CHIP Select

DI : In std\_logic; --Data In

state\_signal : buffer std\_logic\_vector(3 downto 0); --state Out

DO : buffer std\_logic --Data Out

);

signal output\_index:std\_logic\_vector(3 downto 0):="0000";

signal data\_input\_model : std\_logic\_vector(1 downto 0);

signal data : std\_logic\_vector(7 downto 0);

signal receive\_data : boolean:=false;

end ADC0832;

architecture behavior of ADC0832 is

--状态定义

type state is (Start\_Order\_Wait,First\_DI\_Receive,Second\_DI\_Receive,Data\_Transform,Data\_Output);

signal current\_state:state; --现态

signal next\_state:state:=Start\_Order\_Wait; --次态

signal DI1,DI0:std\_logic:='0';

signal output\_order:std\_logic;

begin

--时序电路

synch:process(clk)

begin

--change state with clock

if(clk'event and clk = '0') then

current\_state <= next\_state;

end if;

end process;

--组合电路

state\_trans:process(CS,DI,current\_state)

begin

state\_signal <= "1111";

--work only if CS is low

if(CS = '0') then

case current\_state is

when Start\_Order\_Wait =>

--等待启动命令

-- wait the START BIT

if(DI = '1')then

state\_signal <= "0001";

next\_state <= First\_DI\_Receive;

else

state\_signal <= "0010";

next\_state <= Start\_Order\_Wait;

end if;

output\_order <= '1';

when First\_DI\_Receive =>

--数据通道选择

state\_signal <= "0011";

DI1 <= DI;

next\_state <= Second\_DI\_Receive;

when Second\_DI\_Receive =>

--数据通道选择

state\_signal <= "1000";

DI0 <= DI;

data\_input\_model(1) <= DI1;

data\_input\_model(0) <= DI0;

next\_state <= Data\_Transform;

when Data\_Transform =>

--完成模数转换

state\_signal <= "0101";

case data\_input\_model is

when "00" =>

data<=CONV\_STD\_LOGIC\_VECTOR(CH0-CH1,8);

when "01" =>

data<=CONV\_STD\_LOGIC\_VECTOR(CH1-CH0,8);

when "10" =>

data<=CONV\_STD\_LOGIC\_VECTOR(CH0,8);

when "11" =>

data<=CONV\_STD\_LOGIC\_VECTOR(CH1,8);

end case;

next\_state <= Data\_Output;

output\_order <= '0';

when Data\_Output =>

--串行输出

state\_signal <= "0110";

output\_order <= '0';

next\_state <= Data\_Output;

when others =>

state\_signal <= "1001";

next\_state <= Start\_Order\_Wait;

end case;

else

next\_state <= Start\_Order\_Wait;

end if;

end process;

--数据输出进程

output:process(clk,output\_order)

begin

if(clk'event and clk = '0') then

if(output\_order = '0')then

--先高位在前，再低位在前

case output\_index is

when "0000" =>

DO <= data(7);

output\_index <= output\_index + "0001";

when "0001" =>

DO <= data(6);

output\_index <= output\_index + "0001";

when "0010" =>

DO <= data(5);

output\_index <= output\_index + "0001";

when "0011" =>

DO <= data(4);

output\_index <= output\_index + "0001";

when "0100" =>

DO <= data(3);

output\_index <= output\_index + "0001";

when "0101" =>

DO <= data(2);

output\_index <= output\_index + "0001";

when "0110" =>

DO <= data(1);

output\_index <= output\_index + "0001";

when "0111" =>

DO <= data(0);

output\_index <= output\_index + "0001";

when "1000" =>

DO <= data(1);

output\_index <= output\_index + "0001";

when "1001" =>

DO <= data(2);

output\_index <= output\_index + "0001";

when "1010" =>

DO <= data(3);

output\_index <= output\_index + "0001";

when "1011" =>

DO <= data(4);

output\_index <= output\_index + "0001";

when "1100" =>

DO <= data(5);

output\_index <= output\_index + "0001";

when "1101" =>

DO <= data(6);

output\_index <= output\_index + "0001";

when "1110" =>

DO <= data(7);

output\_index <= output\_index + "0001";

when "1111" =>

DO <= '0';

output\_index<=output\_index;

end case;

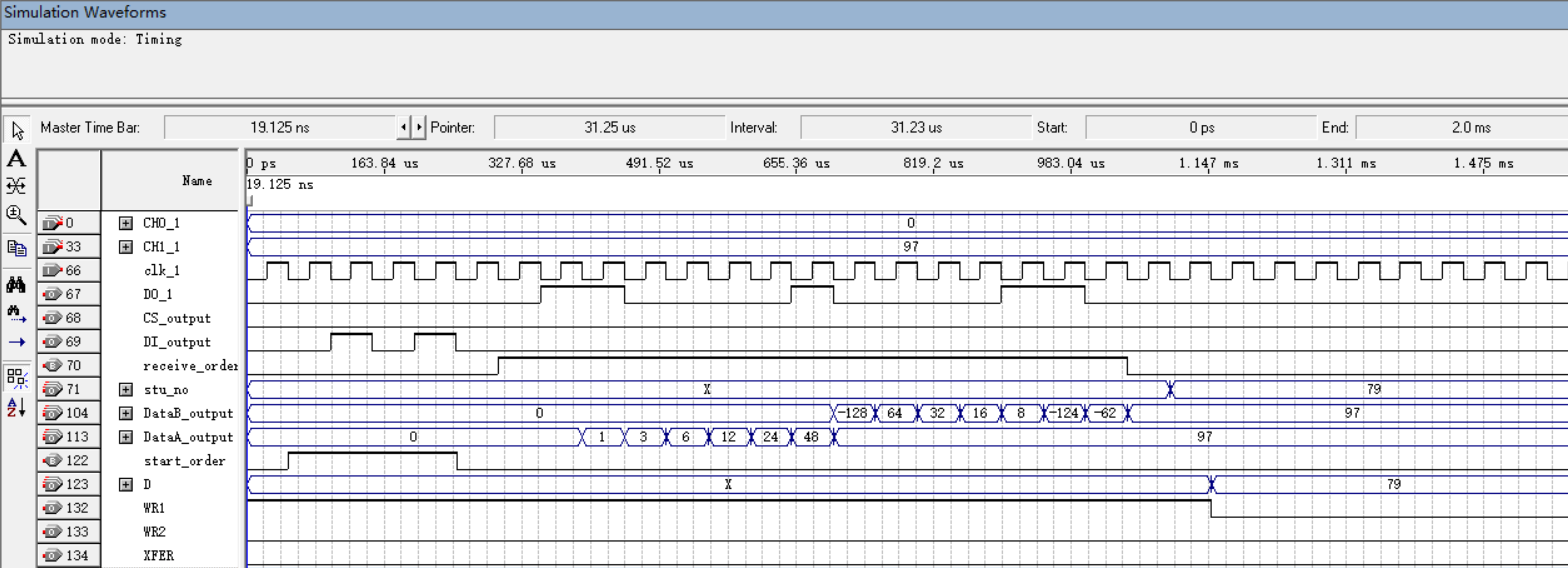
end if;

end if;

end process;

end behavior;

**波形仿真**



控制器进入ADC\_START 状态，start\_order置为高电平，adc\_work进程启动DI\_output依次输出“101”，启动ADC0832并完成通道选择选择，然后进入DATA\_RECEIVE状态，receive\_order置为高电平，receive进程启动，经过15个周期完成数据的接收，然后进入DAC\_START状态，WR1置为低电平，使DAC0832在单缓冲工作模式下开始转换。

**Flow Summary**

****

参考文献

1. 百度文库.ADC0832芯片介绍[DB/OL].<https://wenku.baidu.com/view/15613a8183d049649b665870.html>,2011-10-29/2017-10-9
2. 新浪博客.ADC0832的使用方法[DB/OL].<http://blog.sina.com.cn/s/blog_62b33cb901015zyy.html>,2012-09-11/2017-10-09
3. 吕昕远.AD\_DA.vhd[DB/OL].<https://github.com/pokerfaceSad/DigitalCircuitsCourseDesign/blob/master/AD_DA.vhd>,2017-10-09/2017-10-09
4. 吕昕远.ADC0832.vhd[DB/OL].<https://github.com/pokerfaceSad/DigitalCircuitsCourseDesign/blob/master/ADC0832.vhd>,2017-10-09/2017-10-09