

**FPGA** **课程设计**

**设计(论文)题目基于CycloneII系列芯片的MCU软核实现对直流电机的负反馈调节**

**学院名称 核技术与自动化工程学院**

**专业名称 测控技术与仪器**

**学生姓名 追梦少年南南**

**学生学号 202006010336**

**任课教师 覃章健**

**设计（论文）成绩**

**教务处 制**

2022年12月05日

填写说明

1. 专业名称填写为专业全称，有专业方向的用小括号标明；
2. 格式要求：格式要求：
3. 用A4纸双面打印（封面双面打印）或在A4大小纸上用蓝黑色水笔书写。
4. 打印排版：正文用宋体小四号，1.5倍行距，页边距采取默认形式（上下2.54cm，左右2.54cm，页眉1.5cm，页脚1.75cm）。字符间距为默认值（缩放100%，间距：标准）；页码用小五号字底端居中。
5. 具体要求：

**题目**（二号黑体居中）；

**摘要**(“摘要”二字用小二号黑体居中，隔行书写摘要的文字部分，小4号宋体)；

**关键词**（隔行顶格书写“关键词”三字，提炼3-5个关键词，用分号隔开，小4号黑体)；

正文部分采用三级标题；

**第1章** ××(小二号黑体居中，段前0.5行)

**1.1** ×××××小三号黑体×××××（段前、段后0.5行）

**1.1.1**小四号黑体（段前、段后0.5行）

**参考文献**（黑体小二号居中，段前0.5行），参考文献用五号宋体，参照《参考文献著录规则（GB/T 7714－2005）》。

摘 要

本文详细描述了从零开始制作FPGA软核MCU，实现包括但不限于PIO、定时器、串口等功能，并且通过Nios端写代码的方式实现对整个项目功能的实现。在本设计中，除了相应的软件代码编写，还要加上一定的硬件连接。相应的硬件连接及软件设计在文中都有详细介绍。本项目主要是通过温度传感器以及PWM控制电机实现控制在不同温度下风扇的转速，思想较为简单，但在本人学习过程中具有里程碑的意义。

在实际应用中，无论是处在炎炎夏日的人们还是在不断工作的电脑，都面临着过热的问题。热了就要降温，打开风扇则是降温的不二之选。本项目可以在人体感受不同温度的情况下，进行不同程度的风扇档位调节，该效果同样可以用在电脑风扇转速调节，并成功实现。

**关键词：温度传感器；MCU软核；NiosII；Quartus**

目录

[摘 要 3](#_Toc124778036)

[第一章 引言 5](#_Toc124778037)

[1.1 FPGA技术介绍 5](#_Toc124778038)

[1.2实习内容及要求 5](#_Toc124778039)

[第二章 Quartus端设计 6](#_Toc124778040)

[2.1 MCU制作 6](#_Toc124778041)

[2.1.1 CPU选择 6](#_Toc124778042)

[2.1.2 PIO端口选择 9](#_Toc124778043)

[2.1.3定时器选择 10](#_Toc124778044)

[2.2 编译及调试 10](#_Toc124778045)

[2.2.1 编译结果 10](#_Toc124778046)

[2.2.2 整体结构 12](#_Toc124778047)

[第三章 Nios端设计 13](#_Toc124778048)

[3.1 Nios端基础配置 13](#_Toc124778049)

[3.1.1 新建工程项目 13](#_Toc124778050)

[3.1.2 配置环境变量 13](#_Toc124778051)

[3.2 温度传感器部分 14](#_Toc124778052)

[3.2.1 DS18B20时序 14](#_Toc124778053)

[3.2.2 结果转换 16](#_Toc124778054)

[3.2.3 代码编写 16](#_Toc124778055)

[3.3 PWM控制部分 18](#_Toc124778056)

[3.3.1 控制思想 18](#_Toc124778057)

[3.3.2 代码编写 18](#_Toc124778058)

[3.4 整体代码结构 19](#_Toc124778059)

[3.4.1 整体代码编写 19](#_Toc124778060)

[3.4.2 编译及调试 23](#_Toc124778061)

第一章 引言

1.1 FPGA技术介绍

FPGA（Field Programmable Gate Array）是在PAL （可编程阵列逻辑）、GAL（通用阵列逻辑）等可编程器件的基础上进一步发展的产物。它是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。

在我们的学习中，我们使用到了Quartus8.1。Quartus design 是最高级和复杂的，用于system-on-a-programmable-chip (SOPC)的设计环境。 Quartus design 提供完善的 timing closure 和 LogicLock™ 基于块的设计流程。Quartus design是唯一一个包括以timing closure和基于块的设计流为基本特征的programmable logic device (PLD)的软件。Quartus设计软件改进了性能、提升了功能性、解决了潜在的设计延迟等，在工业领域率先提供FPGA与mask-programmed devices开发的统一工作流程。

我们同样用到了NiosII进行开发。Nios嵌入式处理器，于2001年首次推出，创新的Nios&reg; 嵌入式处理器成为业界第一款专门针对FPGA的商用处理器。自此以后，众多的FPGA用户采用了Altera提供的Nios和Nios II处理器。

1.2实习内容及要求

本次实验要求学生基于Quartus和Nios两款软件制作一个小项目，项目内容由学生自拟。可单独使用Quartus软件，亦可Quartus和Nios联合使用。

第二章 Quartus端设计

2.1 MCU制作

2.1.1 CPU选择

在Quartus8.1软件中，内置了SOPC Builder软件。该软件的主要功能为制作一款MCU。而MCU则分为负责运算的CPU和负责控制外设的外部接口电路。在此我们主要说一下如何先将CPU制作出来。

打开SOPC Builder，界面如图所示：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2.1 SOPC Builder界面

值得强调的是，该图为本人制作完成后的图，与新建项目的图片略有差异。接下来，我们就可以制作相应的CPU。

双击“Nios II Processor”，即可弹出CPU编辑界面。通过自己的相关配置，最终现将CPU模块制作完成。在此，我全部保持默认即可。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2.2 CPU编辑界面

下一步则是RAM的编辑。我们在此不仅需要将RAM制作出来，还要将其与CPU进行关联。而编辑RAM的选项卡则在On-Chip Memory中。在此，我们将RAM设置为32位字节宽，总大小为10240字节。其余部分保持默认即可。

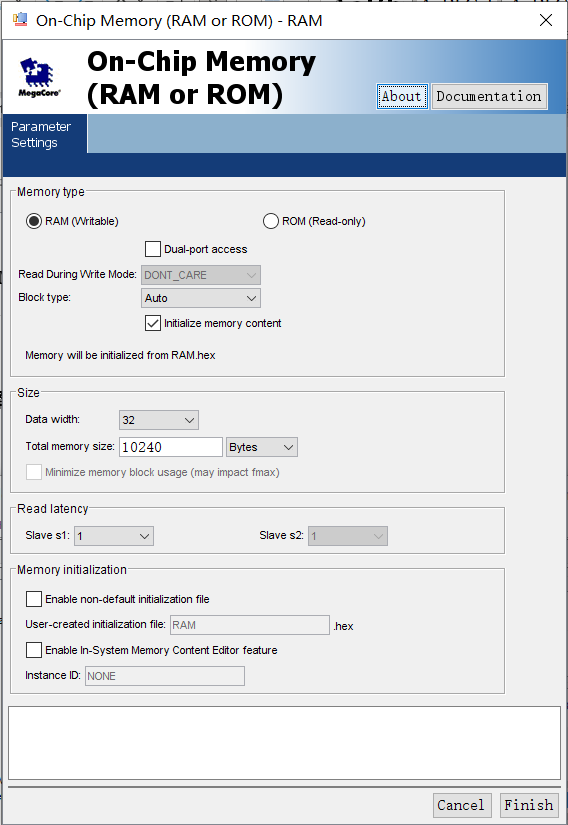


图2.3 RAM编辑界面

最后是JTAG的选择。我们的制作要保证其完整性，调试接口是必不可少的。JTAG作为具有调试功能的接口，我们务必将其考虑在内。JTAG在JTAG to Avalon Master Bridge选项卡下，保持默认选项即可。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2.4 JTAG编辑界面

一切完成就绪后，我们初步的CPU就算是大功告成了。在此，我们添加一下系统时钟sys\_clk和复位按键reset，就可以开始下一步的外设添加了。

2.1.2 PIO端口选择

完成一个项目PIO是不可少的。在本项目中，我们需要一组PIO去读写温度传感器的时序，另外需要一组PIO去控制电机的转动。在SOPC Builder中直接添加即可。另外，为了可以使得芯片更加的充分利用，我们选择在原来的基础上多加几组PIO。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2.5 PIO编辑界面

2.1.3定时器选择

定时器是一个软核MCU中不可或缺的功能，其作用与重要性不言而喻。在软件中，我选择了制作两个定时器：Timer0和Timer1。其中，Timer0用于控制PWM，Timer1用于计时读取温度传感器的值。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图2.6 定时器编辑界面

至此，对整个MCU的编辑已经全部完成。我在此基础上还添加了一组串口用于调试。

2.2 编译及调试

2.2.1 编译结果

编译结果分为两部分：一部分是在SOPC Builder下的MCU编译，另一部分是在Quartus端的整体编译。首先我们来看一下在SOPC Builder下的MCU编译。

在最下面有一个Generate按钮，单机此按钮即可进行编译。若编译成功则会生成，编译失败则会报错。

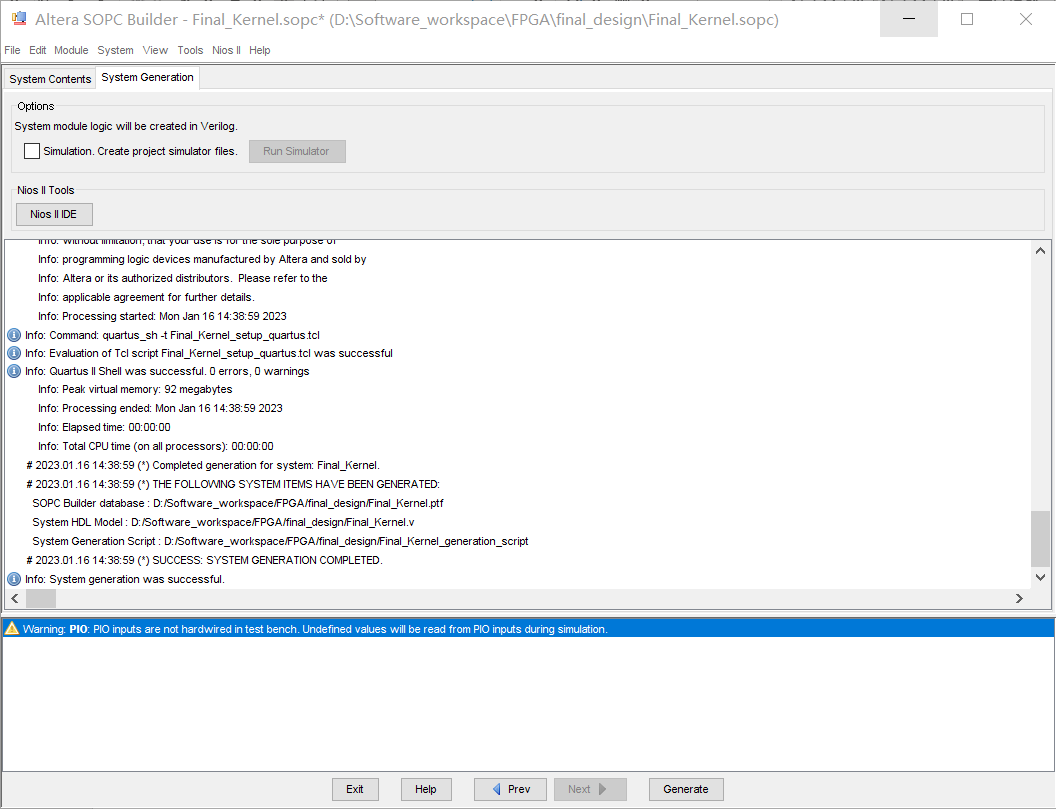


图2.7 在SOPC Builder下的MCU编译

在此，我们看到了“SUCCESS”，证明我们的项目编译成功。接下来则是把它放在Quartus中，放置端口并且进行编译。

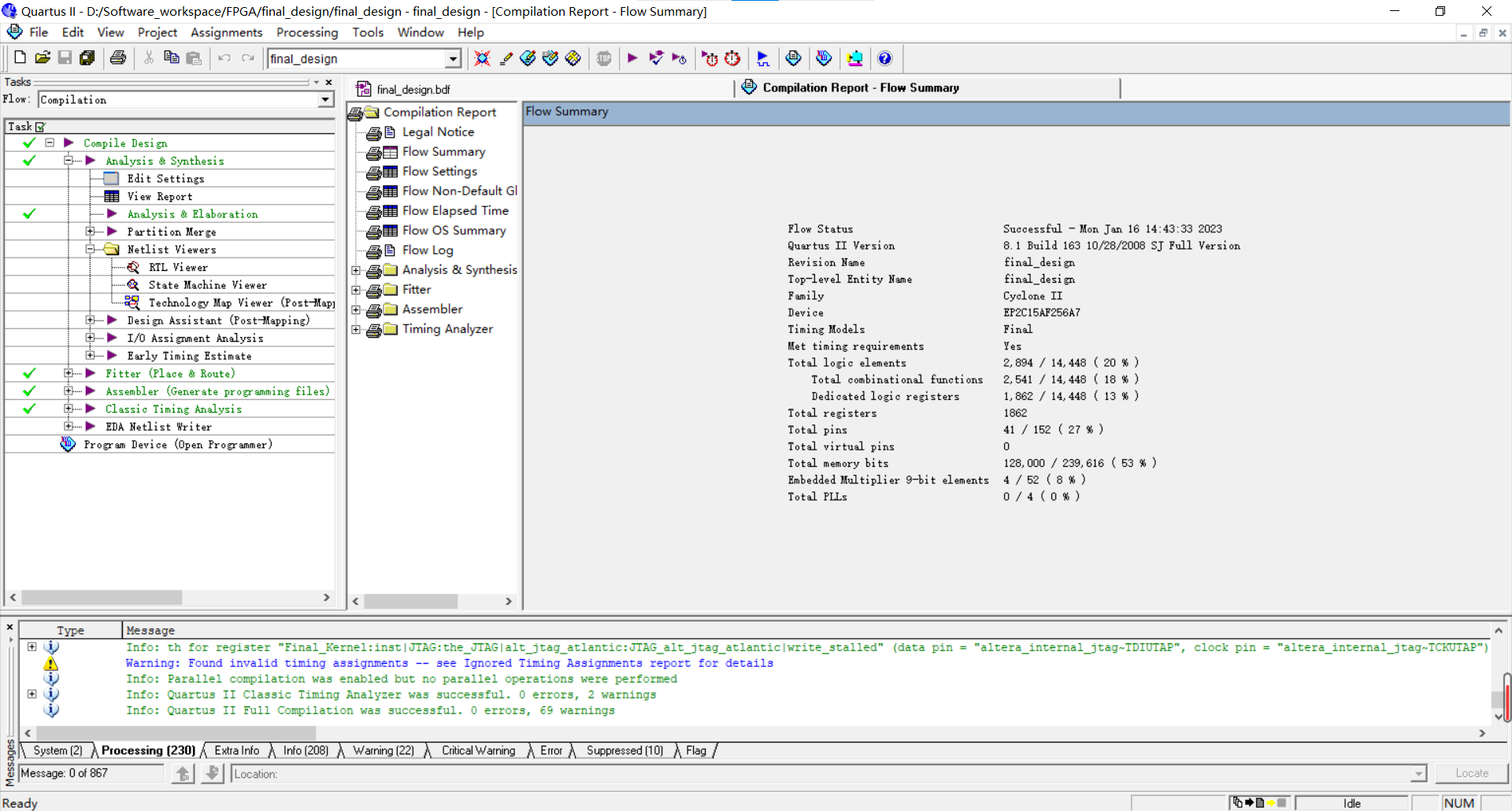


图2.7 Quartus下的MCU编译

可以看到，虽然有一部分警告，但还是编译成功了。接下来我们就要用到编译成功后的文件，就是Nios端的事情了。

2.2.2 整体结构

如图所示为整个MCU的整体结构，包含了16个常规PIO口，一个温度传感器专用接口，一个串口，以及内置了两个定时器。至此，Quartus端全部完成。

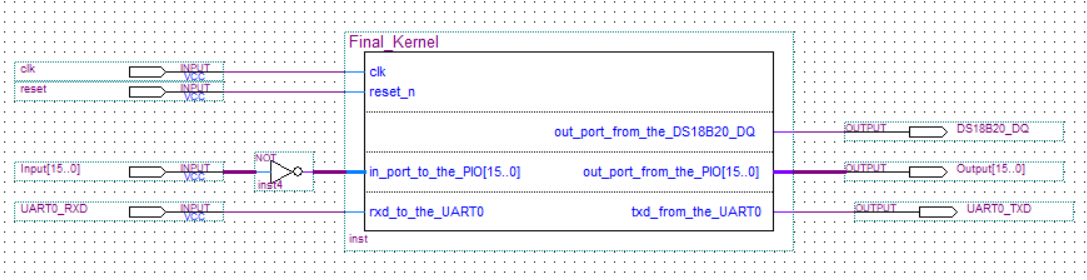


图2.8 整体结构

第三章 Nios端设计

3.1 Nios端基础配置

3.1.1 新建工程项目

在Nios端新建工程项目相对来讲比较容易。偶尔会遇到一些问题也可以在网上寻找解决办法。首先，我们启动Nios软件，在上方的新建项目中找到第二个选项卡并进入，则会得到下图所示界面。在此界面中，编辑你的项目内容，在选择PTF文件时选择你在上一步编译生成的PTF文件，选择OK即可生成两个与你的项目名称一样的项目，一个后面没有“syslib”，一个则有“syslib”。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图3.1 新建工程向导

3.1.2 配置环境变量

如果没有Quartus的环境变量以及jdk的环境变量，Nios是无法使用的。此时，在“我的电脑——属性——高级系统设置——环境变量”中添加一些环境变量如下图所示，即可成功编译Nios的项目。

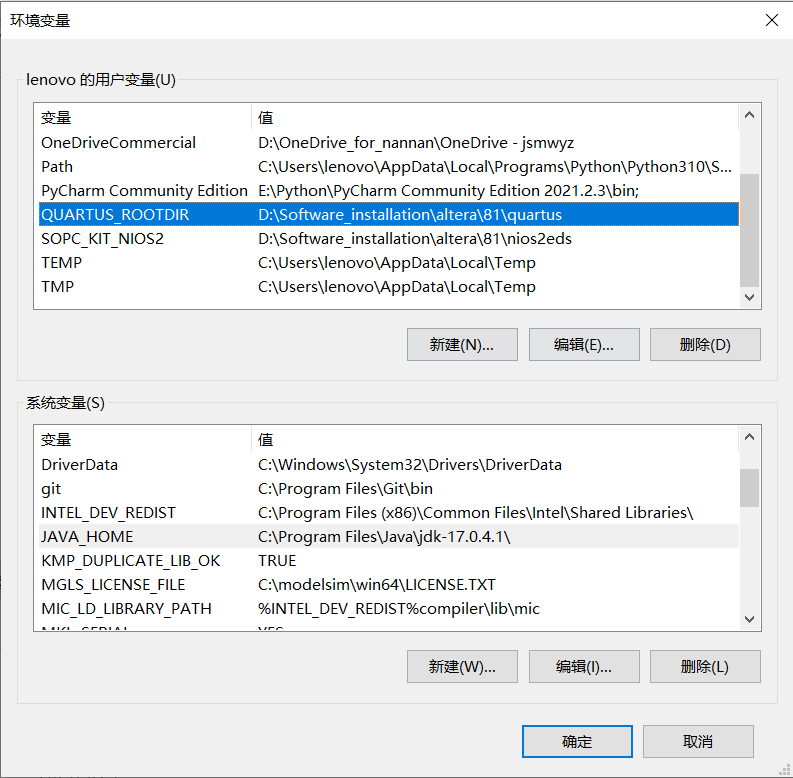


图3.2 环境变量的配置

配置完成你的环境变量时，你可以将你的main函数进行编译，如果编译结果正确，那么恭喜你可以进行下一步的操作了。

3.2 温度传感器部分

3.2.1 DS18B20时序

在本项目中，我选择使用DS18B20温度传感器进行温度检测。DS18B20采用的是one wire总线协议进行读写的，而one wire总线协议则遵照下图所示。

图示

描述已自动生成

图3.3 单总线协议的时序图

根据单总线协议以及我们所查阅的DS18B20芯片手册，得出DS18B20的读写时序图。其中，Reset和Presence可以用一个操作代替，即重置DS18B20.

表格

描述已自动生成

图3.4 DS18B20的读写时序图

3.2.2 结果转换

DS18B20的读取分为两部分，分别是高八位（HSB）和低八位（LSB），而在编写代码时要注意先读高八位后读低八位。

图形用户界面, 文本, 应用程序

描述已自动生成

图3.5 DS18B20的HSB和LSB

在此基础上，读到的数字量也要进行一定的转化，转化过程如图所示。

表格

描述已自动生成

图3.6 DS18B20的数据转化

3.2.3 代码编写

了解了以上部分，我们就可以完成代码的编写了。首先是进行单总线协议的编写。此处我们用到了两个函数分别为操作端口寄存器方向函数以及读写端口输出电平函数。

1. alt\_u8 Reset\_DS18B20(void)

2. {

3. alt\_u8 Presence;

4. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

5. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

6. usleep(650);

7. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

8. usleep(40);

9. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,IN);

10. usleep(40);

11. Presence=IORD\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE)&0x01;

12. usleep(500);

13. return Presence;

14. }

15. void Write\_DS18B20(alt\_u8 Data)

16. {

17. alt\_u8 i;

18. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

19. for(i=0;i<8;i++)

20. {

21. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

22. uDelay(15);

23. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,Data);

24. uDelay(45);

25. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

26. Data>>=1;

27. uDelay(5);

28. }

29. usleep(50);

30. }

31. alt\_u8 Read\_DS18B20(void)

32. {

33. alt\_u8 i,Data=0;

34. for(i=0;i<8;i++)

35. {

36. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

37. Data>>=1;

38. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

39. uDelay(10);

40. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

41. uDelay(1);

42. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,IN);

43. uDelay(2);

44. if((IORD\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE)&0x01)==0x01)

45. Data|=0x80;

46. usleep(10);

47. }

48. return Data;

49. }

接下来是读取温度的函数，严格按照读取时序图进行编写。

1. alt\_u16 Read\_Temperature(void)

2. {

3. alt\_u8 Temp1,Temp2;

4. alt\_u16 t;

5. if(Reset\_DS18B20())

6. return -1;

7. Write\_DS18B20(0xcc);

8. Write\_DS18B20(0x44);

9. Reset\_DS18B20();

10. Write\_DS18B20(0xcc);

11. Write\_DS18B20(0xBE);

12. Temp1=Read\_DS18B20();

13. Temp2=Read\_DS18B20();

14. t=Temp2;

15. t<<=8;

16. t+=Temp1;

17. return t;

18.

19. }

最后是主函数中的读取结果转换。

1. temp=Read\_Temperature();

2. if(temp&0x8000)

3. {

4. temp=~temp;

5. temp=temp+1;

6. }

7. ftemp=(temp\*0.0625)\*100;

8. tempz=ftemp/100;

对结果进行处理后，温度读取结果存储在tempz变量中。

3.3 PWM控制部分

3.3.1 控制思想

PWM即脉冲宽度调制。例如在一个周期中，有50%的时间为高电平，另外50%的时间为低电平时，则可视为此PWM有50%的占空比。PWM可以控制直流电机、舵机等一系列电机，配合温度传感器可以形成闭环控制。

在一个周期time=500内，假设占空比duty=250。我们设计一个count变量从0每隔1ms增加一次，直到增加到time=500。那么令count<duty的时间内，PIO输出高电平，否则输出低电平，这样就完成了一个PWM的脉宽调制。

3.3.2 代码编写

1. static void ISR\_timer0(void\* context,alt\_u32 id)//定时器0中断服务

2. {

3. count++;

4. if(count>time)

5. {count=0;}

6. if(count<=duty)

7. {

8. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(PIO\_BASE,OUT);

9. IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_BASE,1);

10. }

11. else

12. {

13. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(PIO\_BASE,OUT);

14. IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_BASE,0);

15. }

16. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER0\_BASE,0x00);//清空中断标志寄存器

17. }

18. void timer0\_init(void)

19. {

20. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER0\_BASE,0x00);//清空中断标志寄存器

21. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODL(TIMER0\_BASE,1000);//清空中断标志寄存器

22. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODH(TIMER0\_BASE,1000>>16);

23. IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER0\_BASE,0X07);//允许timer0中断

24. alt\_irq\_register(TIMER0\_IRQ,(void\*)TIMER0\_BASE,ISR\_timer0);//注册timer0中断

25. }

26.

3.4 整体代码结构

3.4.1 整体代码编写

1. #include <stdio.h>

2. #include <sys/unistd.h>

3. #include <io.h>

4. #include <string.h>

5. #include "system.h"

6. #include "altera\_avalon\_pio\_regs.h"

7. #include "altera\_avalon\_timer\_regs.h"

8. #include "altera\_avalon\_uart\_regs.h"

9. #include "altera\_avalon\_uart.h"

10. #include "alt\_types.h"

11. #include "sys/alt\_irq.h"

12. #define OUT 1

13. #define IN 0

14. alt\_u16 temp,tempz,tempx;

15. float ftemp;

16. alt\_u16 t10ms;

17. alt\_u16 t500ms;

18. alt\_u16 count=0;//计数

19. alt\_u16 duty=250;//占空比

20. alt\_u16 time=500;//周期

21. static void timer0\_init(void);

22. static void timer1\_init(void);

23. void uDelay(alt\_u32 us)

24. {

25. alt\_u8 i;

26. while(us--)

27. for(i=0;i<2;i++);

28. }

29. alt\_u8 Reset\_DS18B20(void)

30. {

31. alt\_u8 Presence;

32. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

33. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

34. usleep(650);

35. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

36. usleep(40);

37. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,IN);

38. usleep(40);

39. Presence=IORD\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE)&0x01;

40. usleep(500);

41. return Presence;

42. }

43. void Write\_DS18B20(alt\_u8 Data)

44. {

45. alt\_u8 i;

46. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

47. for(i=0;i<8;i++)

48. {

49. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

50. uDelay(15);

51. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,Data);

52. uDelay(45);

53. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

54. Data>>=1;

55. uDelay(5);

56. }

57. usleep(50);

58. }

59. alt\_u8 Read\_DS18B20(void)

60. {

61. alt\_u8 i,Data=0;

62. for(i=0;i<8;i++)

63. {

64. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,OUT);

65. Data>>=1;

66. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,0);

67. uDelay(10);

68. IOWR\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE,1);

69. uDelay(1);

70. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(DS18B20\_DQ\_BASE,IN);

71. uDelay(2);

72. if((IORD\_ALTERA\_AVALON\_PIO\_DATA(DS18B20\_DQ\_BASE)&0x01)==0x01)

73. Data|=0x80;

74. usleep(10);

75. }

76. return Data;

77. }

78. alt\_u16 Read\_Temperature(void)

79. {

80. alt\_u8 Temp1,Temp2;

81. alt\_u16 t;

82. if(Reset\_DS18B20())

83. return -1;

84. Write\_DS18B20(0xcc);

85. Write\_DS18B20(0x44);

86. Reset\_DS18B20();

87. Write\_DS18B20(0xcc);

88. Write\_DS18B20(0xBE);

89. Temp1=Read\_DS18B20();

90. Temp2=Read\_DS18B20();

91. t=Temp2;

92. t<<=8;

93. t+=Temp1;

94. return t;

95.

96. }

98. static void ISR\_timer0(void\* context,alt\_u32 id)//定时器0中断服务

99. {

100. count++;

101. if(count>time)

102. {count=0;}

103. if(count<=duty)

104. {

105. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(PIO\_BASE,OUT);

106. IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_BASE,1);

107. }

108. else

109. {

110. IOWR\_ALTERA\_AVALON\_PIO\_DIRECTION(PIO\_BASE,OUT);

111. IOWR\_ALTERA\_AVALON\_PIO\_DATA(PIO\_BASE,0);

112. }

113. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER0\_BASE,0x00);//清空中断标志寄存器

114. }

115. void timer0\_init(void)

116. {

117. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER0\_BASE,0x00);//清空中断标志寄存器

118. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODL(TIMER0\_BASE,1000);//清空中断标志寄存器

119. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODH(TIMER0\_BASE,1000>>16);

120. IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER0\_BASE,0X07);//允许timer0中断

121. alt\_irq\_register(TIMER0\_IRQ,(void\*)TIMER0\_BASE,ISR\_timer0);//注册timer0中断

122. }

124. static void ISR\_timer1(void\* context,alt\_u32 id)//定时器1中断服务

125. {

126. t10ms++;

127. t500ms++;

128. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER1\_BASE,0x00);//清空中断标志寄存器

129. }

130. void timer1\_init(void)

131. {

132. IOWR\_ALTERA\_AVALON\_TIMER\_STATUS(TIMER1\_BASE,0x00);//清空中断标志寄存器

133. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODL(TIMER1\_BASE,1000);//清空中断标志寄存器

134. IOWR\_ALTERA\_AVALON\_TIMER\_PERIODH(TIMER1\_BASE,1000>>16);

135. IOWR\_ALTERA\_AVALON\_TIMER\_CONTROL(TIMER1\_BASE,0X07);//允许timer1中断

136. alt\_irq\_register(TIMER1\_IRQ,(void\*)TIMER1\_BASE,ISR\_timer1);//注册timer1中断

137. }

139. int main()

140. {

141. timer1\_init();

142. timer0\_init();

143. while(1)

144. {

145. if(t10ms>=10)

146. {

147. t10ms=0;

148. if(t500ms>=10)//每500ms读取一次ds18b20

149. {

150. t500ms=0;

151. temp=Read\_Temperature();

152. if(temp&0x8000)

153. {

154. temp=~temp;

155. temp=temp+1;

156. }

157. ftemp=(temp\*0.0625)\*100;

158. tempz=ftemp/100;

159. }

160. switch(tempz)

161. {

162. case 23:

163. duty=100;break;

164. case 24:

165. duty=200;break;

166. case 25:

167. duty=300;break;

168. case 26:

169. duty=400;break;

170. case 27:

171. duty=500;break;

172. default:

173. duty=0;break;

174. }

175. }

176.

177. }

178. return 0;

179. }

180.

在此代码中，我每隔500ms读取一次DS18B20的值，并通过swtich语句闭环控制duty变量进而直接控制电机转速。电机转速越快，风扇风力越强。风扇风力变强，温度就会降低，风扇风力就会减小，从而实现闭环。

3.4.2 编译及调试

代码编写完成后，我们进行编译，发现编译成功。至此，Nios端的任务已全部完成，整体课程设计全部完成。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图3.7 Nios端编译成功

|  |  |
| --- | --- |
| **学生学习心得** | 在FPGA课程中，我掌握了与单片机不同的开发方式，运用了更多专业性软件进行创作开发，继而达到学以致用的目的。整个FPGA的实验及课程设计由我本人完成，在此过程中我不仅获得了知识，更获得了成功的喜悦。这种喜悦会成为我进一步走下去的动力，敦促我在电子技术这一条路上越走越远。最后我要感谢帮助过我的老师和同学，有了你们才有了我这一完整的实验报告。  学生（签名）：  2022年12月11日 |
| **诚信承诺** | 本人郑重声明所呈交的课程报告是本人在指导教师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同学对本文研究所做的贡献均已在报告中作了明确的说明并表示谢意。  学生（签 名）： |
| **任课**  **教师**  **评语** | 成绩评定：  任课教师（签名）：  年 月 日 |