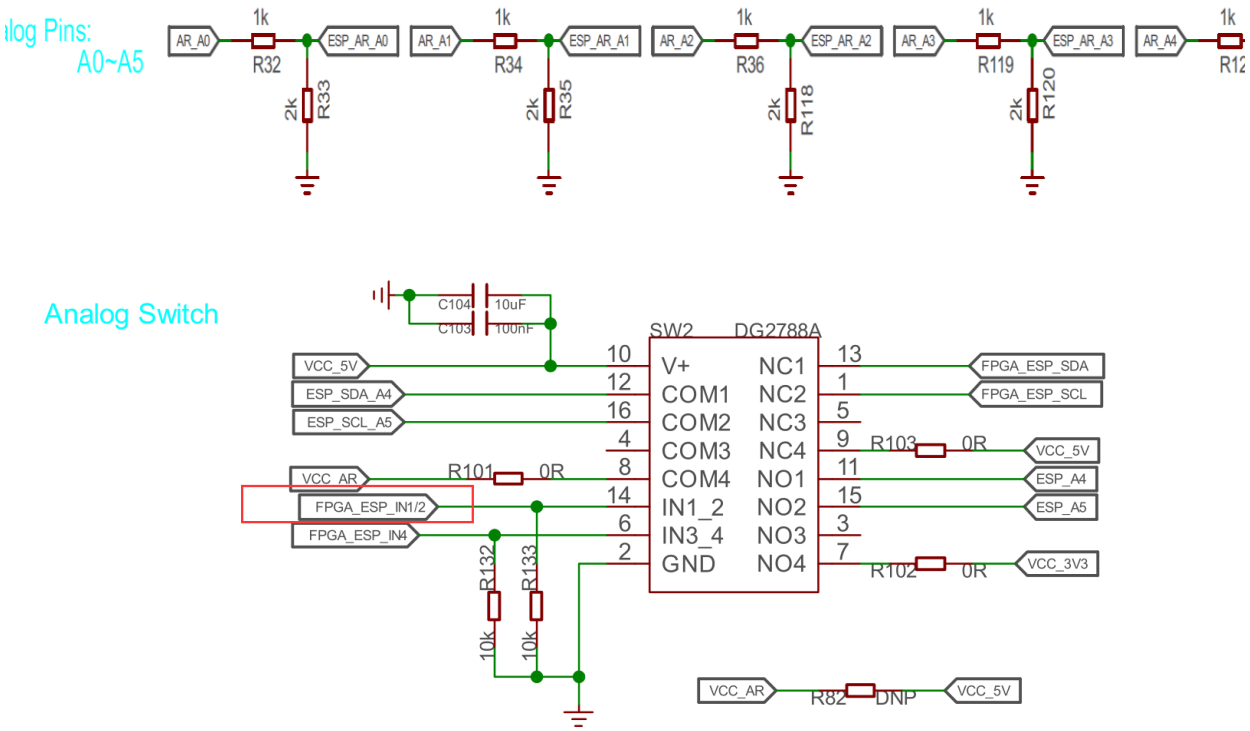
测试记录

ADC 采样：

步骤：esp32 开发板 DAC 输出，测试板 A0~A5 进行 ADC 采样

硬件连线：esp32 DAC 输出端口（esp32DAC 输出引脚为 GPIO25,GPIO26，此处使用 GPIO25）分别连接 AR_A0~ARA5,测试板从 ESP_AR_A0~ ESP_AR_A5 读取电压值，测得电压值为 DAC 输出电压的 2/3



注：FPGA 这个引脚需要输出高电平才能将测试板上 esp32 A4/A5 引脚用于 ADC 采样

程序：esp32 小板子：

```
import machine
dac=machine.DAC(25)
dac.write(0)          //dac 输出理论值为 0V
dac.write(255)        //dac 输出理论值为 3.3V
dac.write(200)        //dac 输出理论值为 2.58V
```

测试版 ADC：

```
import machine
a0=machine.ADC(36)
a1=machine.ADC(37)
a2=machine.ADC(38)
a3=machine.ADC(39)
a4=machine.ADC(32)
a5=machine.ADC(33)
```

```
//改变 ADC 的输入电压范围,设置为最大采样电压为 3.9V
a0.atten(a0.ATTN_11DB)
a1.atten(a1.ATTN_11DB)
a2.atten(a2.ATTN_11DB)
a3.atten(a3.ATTN_11DB)
a4.atten(a4.ATTN_11DB)
a5.atten(a5.ATTN_11DB)
//读出电压值以 mv 为单位
a1.read()
a2.read()
a3.read()
a4.read()
a5.read()
```

现象：当 ADC 输出 3.3V 时万用表测得输出电压为 2.9V，ADC 理论采样为 1.93V，实际采样为 1920mv 左右

当 ADC 输出 2.58V 时万用表测得输出电压为 2.3V，ADC 理论采样为 1.53V，实际采样为 1530mv 左右

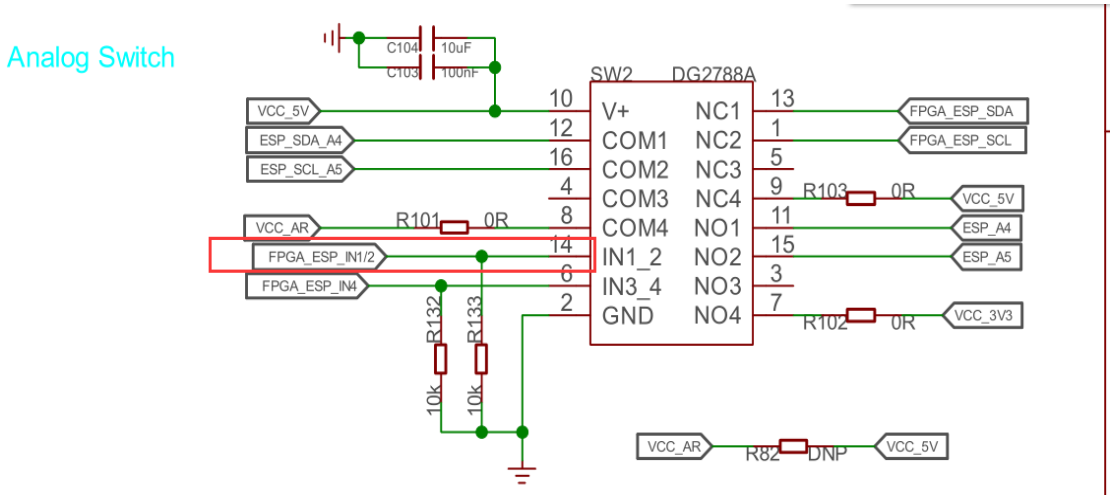
分析：ADC 运行正常

I2C 普通通信:

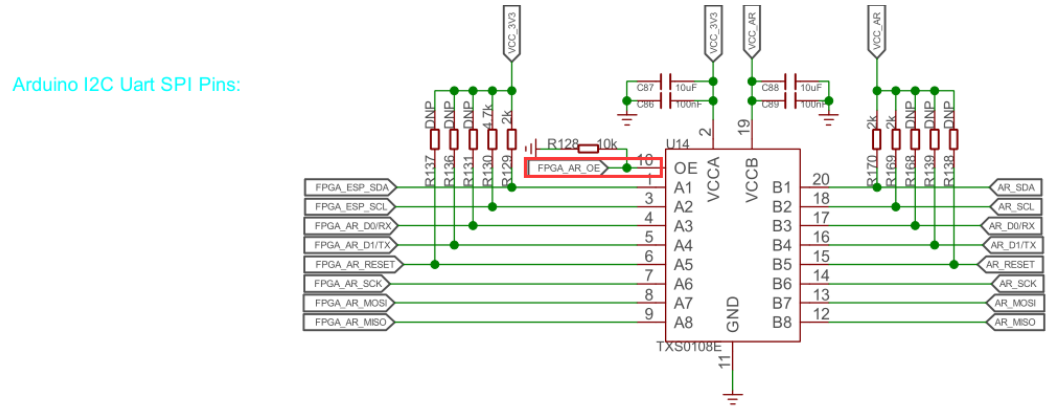
步骤：esp32 作主机，测试板上 esp32 作从机

硬件连线：esp32 上两引脚（sda=21，scl=22）与测试板上两个引脚（AR_SCL、AR_SDA）相连

FPGA 上需要控制两个引脚输出相应电平才可正常进行 I2C 通信



应使 FPGA_ESP_IN1/2 引脚输出低电平(此处虽然默认下拉连上 I2C 但是测得的是连在 ADC)



应使 FPGA_AR_OE 输出高电平，此处默认的是低电平

程序：esp32 板子(主机)：

```
import machine
m = machine.I2C(0, sda=21, scl=22, speed=400000)
m.scan()  //获取 I2C 上所有设备地址
 m. is_ready(32) //判断地址为 32 的从机是否准备好(ESP32 地址默认为 32,可修改)
m.writeto_mem(32, 40, "Hi from master")//测试板上默认划分了 256 字节（可设置大小）的缓冲区，此处是将"Hi from master"写入了缓冲区中地址为 40 开头的区域
m.readfrom_mem(32,40,14,stop=False)  //读取测试板缓冲区地址为 40 开头的内容
```

现象：

```
[32, 107]
>>> m.scan()
[32, 107]
>>> m.scan()
[32, 107]
```

32 为测试板 esp32 默认地址，107 为板上 LSM6DS3TR 地址

```
>>> m.writeto_mem(32, 40, "Hi from master")
14
```

返回成功写字节数

```
>>> m.readfrom_mem(32, 40, 14, stop=False)
b'Hi from master'
```

返回读出内容

测试板 esp32(从机)：

```
import machine
//回调函数：esp322 主机发送、借收、主机设置从机缓冲区地址时会执行相应的程序
//分别是 machine.I2C.CBTYPE_TXDATA（发送）
//       machine.I2C.CBTYPE_RXDATA（接收）
//       machine.I2C.CBTYPE_ADDR（主机设置从机缓冲区地址）
//此处是在这三种情况下输出相应提示信息
def my_task(res):
    cbtype = res[0]
    if cbtype == machine.I2C.CBTYPE_TXDATA:
        print("ESP32 Data sent to Arduino : addr={}, len={}, ovf={},
data={}".format(res[1], res[2], res[3], res[4]))
    elif cbtype == machine.I2C.CBTYPE_RXDATA:
        print("ESP32 Data received from master: addr={}, len={}, ovf={},
data: [{}]".format(res[1], res[2], res[3], res[4]))
    elif cbtype == machine.I2C.CBTYPE_ADDR:
        print("ESP32 Addres set: addr={}".format(res[1]))
    else:
        print("Unknown CB type, received: {}".format(res))

s = machine.I2C(1, mode=machine.I2C.SLAVE, sda=32, scl=33)
//回调函数配置
s.callback(my_task, s.CBTYPE_ADDR | s.CBTYPE_RXDATA | s.CBTYPE_TXDATA)
//查看 ADDR 从 40 开始的 14 个数据
s.getdata(40, 14)
```

现象：

主机执行完 `m.writeto_mem(32, 40, "Hi from master")`后

从机输出提示信息：

```
>>> ESP32 Data received from master: addr=40, len=14, ovf=0, data: [b'Hi from ma
ster']
```

主机执行完 m.readfrom_mem(32,40,14,stop=False)

从机输出提示信息：

```
>>> ESP32 Data received from master: addr=40, len=14, ovf=0, data: [b'Hi from ma
ster']
```

从机执行 `s.getdata(40, 14)`后

从机输出提示信息：

```
>>> m.readfrom_mem(32, 40, 14, stop=False)
b'Hi from master'
```

测试板上 esp32 作主机，外接 esp32 板作从机测试（将程序反过来用即可）

测试版 I2C 主机正常可用

//Arduino esp32 FPGA I2C 通信烧程序

步骤: Arduino 作主机，测试板上 esp32 作从机，收到相应指令后，对 FPGA 进行程序烧写

硬件连线: Arduino 和测试板接在一起

通信格式:

地址	缓冲区（寄存器）地址	数据
32（默认）测试板 esp32 如修改则按修改过的	此处是定义的 0—4（可扩展）	

支持连续发送多字节（发一次从机地址，寄存器地址，随后连续发多个数据给连续的寄存器赋值）

目前程序功能只设定了发送后给 FPGA 烧写程序的指令（寄存器地址为 1）
发送的指令格式为：
发送从机地址 32 和起始位，writebyte(1)，writebyte(num)，发送停止位
Num 是要烧写的文件编号

程序：

测试板

```
import os
import time
import json
import machine
from machine import Pin
import xfpga

boot_btn = Pin(0,mode=Pin.IN)
if(boot_btn()):
  start = time.ticks_ms()
  xfpga.overlay('esp32.bit')
  print(time.ticks_ms()-start)

f_cfg=open("/sd/board_config.json")
Board_Config=json.loads(f_cfg.read())
OverList=Board_Config["Overlay_List"]
print(OverList)


#I2C and cmd
```

```
//回调函数：输出提示信息和命令判断与执行
def my_task(res):
    global OverList
    cbtype = res[0]
    #display when slave send to master
    if (cbtype == machine.I2C.CBTYPE_TXDATA):
        print("ESP32 Data sent to Arduino : addr={}, len={}, ovf={},
data={}".format(res[1], res[2], res[3], res[4]))
    #display when slave receive from master
    elif (cbtype == machine.I2C.CBTYPE_RXDATA):
        print("ESP32 Data received from master: addr={}, len={},
ovf={}, data: [{}]".format(res[1], res[2], res[3], res[4]))
        cmd = s.getdata(0, 5)
        #ADD YOUR  CODE (CMD)  HERE
        if(cmd[0] == 97 ):#a
          print("cmd0")
          s.setdata('0',0)
        if(cmd[1] in range(0,len(OverList))):#d
          xfpga.overlay(OverList[cmd[1]])
          s.setdata('0',1)
        if(cmd[2] == 101):#e
          print("cmd2")
          s.setdata('0',2)
        if(cmd[3] == 102):#f
          print("cmd3")
          s.setdata('0',3)
        if(cmd[4] == 103):#g
          print("cmd=4")
          s.setdata('0',4)
s = machine.I2C(1, mode=machine.I2C.SLAVE, sda=32, scl=33)
s.callback(my_task, s.CBTYPE_ADDR | s.CBTYPE_RXDATA |
s.CBTYPE_TXDATA)
```

Arduino 程序

```
#include<Wire.h>
void setup() {
  // put your setup code here, to run once:
Wire.beginTransmission(32);        //发送从机地址和起始位
Wire.write(0x01);                  //发送寄存器地址
Wire.write(0);                     //发送数据
Wire.endTransmission();            //发送停止位
```

```
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

现象：Arduino 发送指定数据后 ESP32 给 FPGA 下载相应程序