APB-I2C

2024.08.28

一 修订

版本	日期	编辑人	内容
1.00	2024.08.28	陈家耀	创建了第一个正式版本

二 简介和特性

APB-I2C 是一个带有 APB 从接口的 I2C 控制器，可直接挂载在 APB 总线上充当 I2C 外设。本 IP 简单易用、资源消耗少，具有以下特性：

- **支持 7 位或 10 位 I2C 地址**
- **运行时可配置的 I2C 时钟分频系数**
- **可启用 I2C 收发中断**

APB-I2C 的组成如图 2-1 所示，它由 **APB 寄存器接口**、**收发 FIFO**、**中断控制**和 **I2C 控制器**四部分组成。在 **I2C 控制器**中，从发送 FIFO 中取出**传输信息包**，开启一段 I2C 传输，分为地址和数据阶段，每个传输字节都包含传输模式（带起始位、带结束位或正常）和传输方向（发送或接收）等信息，由 I2C 主接口代理实现字节级的 I2C 传输。若为 I2C 接收，则将接收到的字节数据存入接收 FIFO。

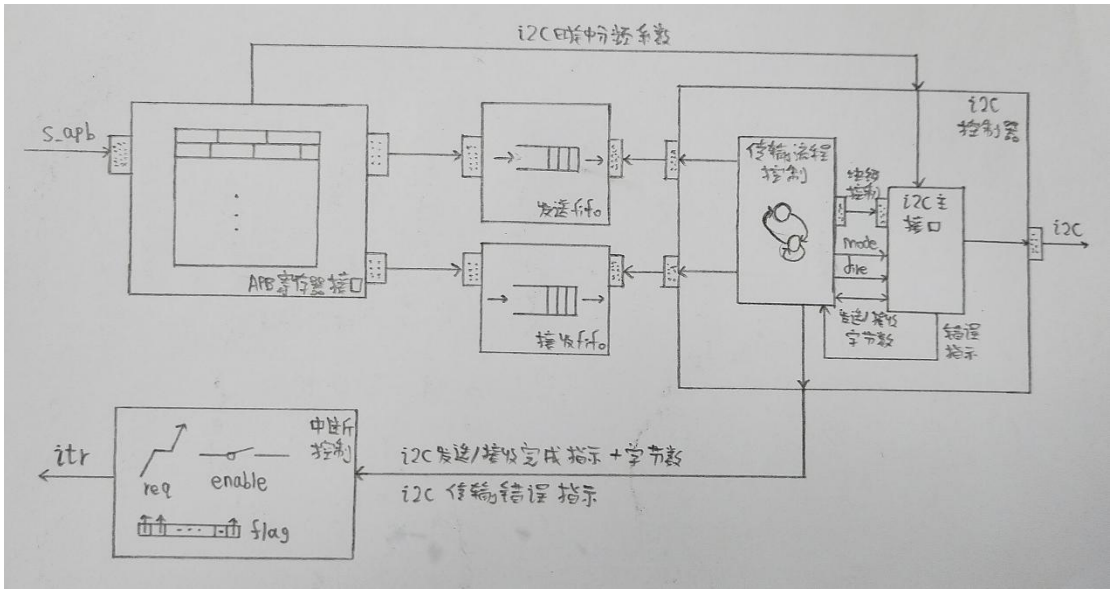

图 2-1 APB-I2C 组成框图

三 IP 功能

APB-I2C 是通用的 I2C 外设，能够驱动 I2C 接口收发数据，可启用 I2C 收发中断。其功能描述如下：

1、**实现 I2C 传输**。用户通过 APB 寄存器接口向发送 FIFO 写入**传输信息包**，I2C 控制器自动从发送 FIFO 取出**传输信息包**，驱动 I2C 接口实现数据传输。传输信息包中的每个字节附带一个 last 信号用于指示包结束，其格式如下：

写数据 -> last0 xxxx_xxx0 地址阶段#0
[last0 xxxx_xxxx 地址阶段#1（仅 10 位地址时需要）]
last0 xxxx_xxxx 数据阶段（数据#0）
...
last1 xxxx_xxxx 数据阶段（数据#N-1）

读数据 -> last0 xxxx_xxx1 地址阶段#0
[last0 xxxx_xxxx 地址阶段#1（仅 10 位地址时需要）]
last1 8 位待读取字节数

若为 I2C 接收传输，I2C 控制器自动向接收 FIFO 存入接收到的字节数据，用户可通过 APB 寄存器接口从接收 FIFO 取出数据。

2、**可配置的 I2C 时钟分频系数**。I2C 时钟分频系数指定了传输数据和停止位时 SCL 的频率。

3、**可启用的 I2C 收发中断**。APB-I2C 支持**发送指定字节数中断**、**接收指定字节数中断**、**从机响应错误中断**和**接收溢出中断**。发送或接收中断字节数可通过 APB 寄存器接口进行配置。

四 IO 描述

表 4-1 APB-I2C IO 表

端口名	方向	位宽	含义
时钟和复位			
clk	input	1	时钟
resetn	input	1	复位，低有效
APB 从接口			
paddr	input	32	APB 从机地址
psel	input	1	APB 从机片选
penable	input	1	APB 从机传输使能
pwrite	input	1	APB 从机读写类型
pwdata	input	32	APB 从机写数据
pready_out	output	1	APB 从机传输完成，固定为 1
prdata_out	output	32	APB 从机读数据
pslverr_out	output	1	APB 从机传输错误，固定为 0
I2C 接口			
scl_t	output	1	SCL 三态方向，1'b1 为输入，1'b0 为输出
scl_i	input	1	SCL 三态输入
scl_o	output	1	SCL 三态输出
sda_t	output	1	SDA 三态方向，1'b1 为输入，1'b0 为输出
sda_i	input	1	SDA 三态输入
sda_o	output	1	SDA 三态输出
中断信号			
itr	output	1	I2C 外设中断请求

五 可配置参数描述

表 5-1 APB-I2C 可配置参数表

配置参数名	含义	可取值
addr_bits_n	I2C 地址位数	7 | 10
en_i2c_rx	是否使能 I2C 接收	"true" | "false"
tx_rx_fifo_ram_type	收发 FIFO 的 RAM 类型	"lutram" | "bram"
tx_fifo_depth	发送 FIFO 深度	32 | 64 | 128 | ... | 2048
rx_fifo_depth	接收 FIFO 深度	32 | 64 | 128 | ... | 2048
simulation_delay	仿真延时，可用于仿真时模拟 D 到 Q 延迟	0.1f~100.0f

六 应用指南

6.1 RTL 设计指南

APB-I2C 是标准的 APB 外设，请将 APB-I2C 挂载在 APB 总线上使用，典型情况是挂载在 AXI-APB 桥或 AHB-APB 桥上作为一个 APB 从机，如图 6-1-1 所示。关于 AXI-APB 桥或 AHB-APB 桥，请参见 UG200。

本 IP 所提供的同步 FIFO 的顶层 RTL 文件为 ram_fifo_wrapper.v，由于 fifo 使用到的 RAM 可能与器件类型有关，必要时请根据设计要求进行替换。

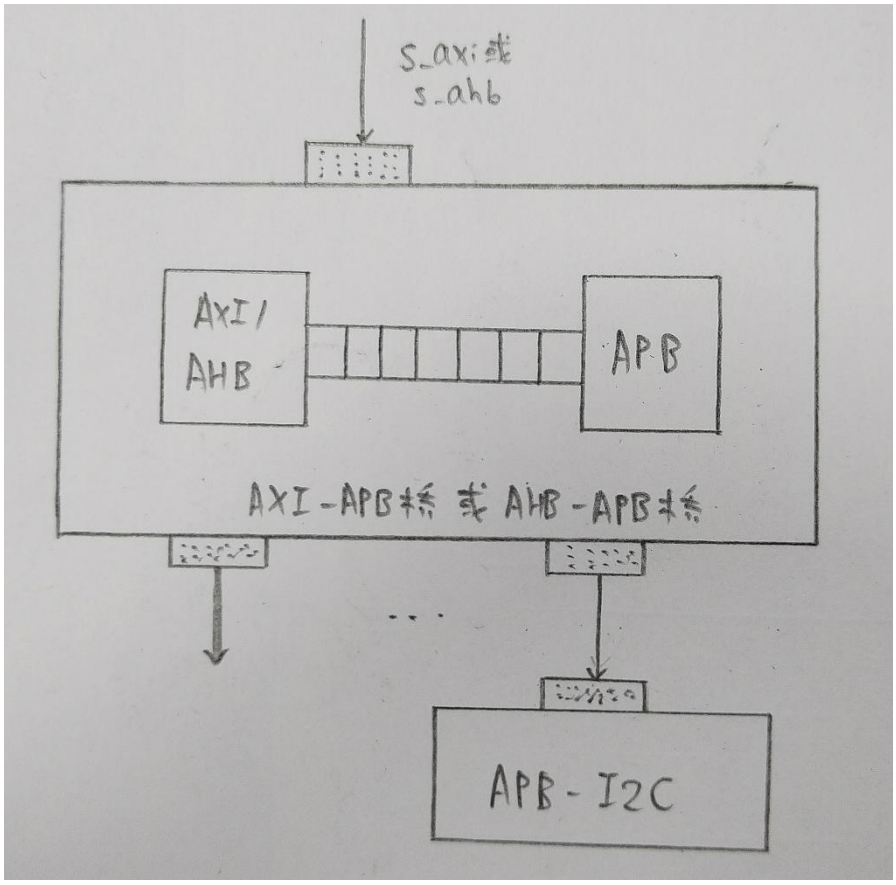

图 6-1-1 APB-I2C 应用图

6.2 软件编程指南

6.2.1 软件驱动 API

1、类型定义

• ApbI2C 结构体（APB-I2C 外设结构体）

ApbI2CHd* hardware：APB-I2C 寄存器接口结构体指针，映射到 I2C 外设的寄存器接口

• ApbI2CHd 结构体（APB-I2C 寄存器接口结构体）

uint32_t fifo_cs：收发 fifo 控制
uint32_t itr_status_en：中断使能
uint32_t itr_th：运行时参数
uint32_t itr_flag_status：中断标志和状态信息

2、宏定义

- 中断类型掩码

I2C_TX_BYTES_N_ITR_MASK：I2C 发送指定字节数中断标志
I2C_SLAVE_RESP_ERR_ITR_MASK：I2C 从机响应错误中断标志
I2C_RX_BYTES_N_ITR_MASK：I2C 接收指定字节数中断标志
I2C_RX_OVF_ITR_MASK：I2C 接收溢出中断标志

3、函数

- void apb_i2c_init(ApbI2C* i2c, uint32_t base_addr);

简介：初始化 APB-I2C
参数：**i2c** APB-I2C（结构体指针）
base_addr APB-I2C 外设基地址
返回值：无

- int apb_i2c_start_wt_trans(ApbI2C* i2c, uint8_t slave_addr, uint8_t* data, uint8_t len);

简介：APB-I2C 启动写传输
参数：**i2c** APB-I2C（结构体指针）
slave_addr 从机地址
data 待发送字节缓冲区
len 待发送字节数
注意：向发送 fifo 写数据可能会产生阻塞
返回值：是否成功

- int apb_i2c_start_rd_trans(ApbI2C* i2c, uint8_t slave_addr, uint8_t len);

简介：APB-I2C 启动读传输
参数：**i2c** APB-I2C（结构体指针）
slave_addr 从机地址
len 待接收字节数
注意：向发送 fifo 写数据可能会产生阻塞
返回值：是否成功

- int apb_i2c_get_rx_byte(ApbI2C* i2c, uint8_t* byte);

简介：从接收 FIFO 获取一个字节数据
参数：**i2c** APB-I2C（结构体指针）
byte 接收字节缓冲区（首地址）
返回值：是否成功

- void apb_i2c_enable_itr(ApbI2C* i2c, uint8_t itr_en);

简介：APB-I2C 使能中断
参数：**i2c** APB-I2C（结构体指针）
itr_en 中断使能向量
返回值：无

- void apb_i2c_disable_itr(ApbI2C* i2c);
  简介：APB-I2C 除能中断
  参数：**i2c** APB-I2C（结构体指针）
  返回值：无

- void apb_i2c_config_params(ApbI2C* i2c, uint8_t tx_bytes_n_th, uint8_t rx_bytes_n_th, uint8_t scl_div_n);
  简介：APB-I2C 配置运行时参数
  参数：**i2c** APB-I2C（结构体指针）
  tx_bytes_n_th I2C 发送中断字节数阈值
  rx_bytes_n_th I2C 接收中断字节数阈值
  scl_div_n I2C 时钟分频系数
  返回值：无

- uint8_t apb_i2c_get_itr_status(ApbI2C* i2c, uint16_t* tx_bytes_n, uint16_t* rx_bytes_n);
  简介：APB-I2C 获取中断状态
  参数：**i2c** APB-I2C（结构体指针）
  tx_bytes_n I2C 发送字节数（指针）
  rx_bytes_n I2C 接收字节数（指针）
  返回值：中断状态

- void apb_i2c_clear_itr_flag(ApbI2C* i2c);
  简介：APB-I2C 清除中断标志
  参数：**i2c** APB-I2C（结构体指针）
  返回值：无

6.2.2 软件编程示例

本示例基于 APB-I2C 实现了 EEPROM 的读写，先向地址 0x00 写入字节 0x0A，向地址 0x01 写入字节 0xFE，然后读取地址 0x00 和 0x01。EEPROM 型号为 AT24CXX。

```
/*****************************************************************************************************************
APB-I2C 示例代码
@brief  基于 APB-I2C 实现 EEPROM 读写
@attention 请根据硬件平台更换与延迟(delay)相关的 API
@date   2024/08/28
@author 陈家耀
@eidt   2024/08/28 1.00 创建了第一个正式版本
```

```
8.      ********************************************************************************************
    ***********************************/
9.
10.     #include "../apb_i2c.h"
11.     #include "../delay.h"
12.
13.     ////////////////////////////////////////////////////////////////////////////////////////////////
14.
15.     #define APB_I2C_BASEADDR 0x40000000 // APB-I2C 外设结构体
16.
17.     #define EEPROM_I2C_ADDR 0xA0 // EEPROM I2C 从机地址
18.
19.     ////////////////////////////////////////////////////////////////////////////////////////////////
20.
21.     static ApbI2C i2c; // APB-I2C 外设基地址
22.
23.     ////////////////////////////////////////////////////////////////////////////////////////////////
24.
25.     static uint8_t read_eeprom_by_i2c(ApbI2C* i2c, uint8_t slave_addr, uint16_t mem_addr); // 读 EEPROM
26.     static void write_eeprom_by_i2c(ApbI2C* i2c, uint8_t slave_addr, uint16_t mem_addr, uint8_t data); //
    写 EEPROM
27.
28.     ////////////////////////////////////////////////////////////////////////////////////////////////
29.
30.     /*************************
31.     @io
32.     @public
33.     @brief  读 EEPROM
34.     @param  i2c APB-I2C(结构体指针)
35.             slave_addr I2C 从机地址
36.                     mem_addr 读取的地址
37.     @return 读取到的字节数据
38.     *************************/
39.     static uint8_t read_eeprom_by_i2c(ApbI2C* i2c, uint8_t slave_addr, uint16_t mem_addr){
40.         uint8_t recv_buf;
41.
42.         uint8_t send_buf[2] = {(uint8_t)(mem_addr >> 8), (uint8_t)(mem_addr & 0x00FF)};
43.
44.         apb_i2c_start_wt_trans(i2c, slave_addr, send_buf, 2); // 启动写传输
45.         apb_i2c_start_rd_trans(i2c, slave_addr, 1); // 启动读传输
46.
47.         while(!apb_i2c_get_rx_byte(i2c, &recv_buf)); // 获取 1 个字节
48.
49.         return recv_buf;
```

```
}

/*************************
@io
@public
@brief  写 EEPROM
@param  i2c APB-I2C(结构体指针)
        slave_addr I2C 从机地址
              mem_addr 写入的地址
            data 待写入的字节数据
@return none
*************************/
static void write_eeprom_by_i2c(ApbI2C* i2c, uint8_t slave_addr, uint16_t mem_addr, uint8_t data){
    uint8_t send_buf[3] = {(uint8_t)(mem_addr >> 8), (uint8_t)(mem_addr & 0x00FF), data};

    apb_i2c_start_wt_trans(i2c, slave_addr, send_buf, 3); // 启动写传输

    delay_ms(10); // 延时 0.01s
}

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

void apb_i2c_eeprom_example(void){
    // 初始化 APB-I2C
    apb_i2c_init(&i2c, APB_I2C_BASEADDR);

    // 配置运行时参数
    // I2C 时钟分频系数 = 7
    apb_i2c_config_params(&i2c, 1, 1, 7);

    // 向地址 0x00 写入字节 0x0A
    write_eeprom_by_i2c(&i2c, EEPROM_I2C_ADDR, 0x00, 0x0A);

    // 向地址 0x01 写入字节 0xFE
    write_eeprom_by_i2c(&i2c, EEPROM_I2C_ADDR, 0x01, 0xFE);

    // 读取地址 0x00 和 0x01
    uint8_t rdata[2];

    rdata[0] = read_eeprom_by_i2c(&i2c, EEPROM_I2C_ADDR, 0x00);
    rdata[1] = read_eeprom_by_i2c(&i2c, EEPROM_I2C_ADDR, 0x01);

    if((rdata[0] == 0x0A) && (rdata[1] == 0xFE)){
        // 读写验证成功
```

```
94.             // ...
95.         }else{
96.             // 读写验证失败
97.             // ...
98.         }
99.
100.        while(1);
101.    }
```