# EEPROM 控制模块说明

## 功能描述

本模块的功能为EEPROM – AT24LC04 (暂定)芯片的读写控制。

1. 写操作：根据输入的数据(32bit)和地址(7bit)以及控制位(1bit)，将数据从低8位开始写入EEPROM。以Page write的方式实现。地址后补2’b00作为第一个byte的地址。
2. 读操作：根据输入的地址(7bit)和 控制位(1bit)，从EEPROM中读出4 byte数据，并按照先读出的数据放在低8位的顺序排列成32bit数据输出。以sequential read的方式实现。地址后补2’b00作为第一个byte的地址。

## 引脚描述

模块引脚描述如表1.

表1

|  |  |  |  |
| --- | --- | --- | --- |
| PIN | 方向 | 位数 | 功能 |
| clk | input | 1 | 模块时钟信号 |
| rst\_n | input | 1 | 模块复位信号 |
| scl | output | 1 | IIC时钟信号 |
| sda | inout | 1 | IIC数据信号 |
| ctrl\_byte | input | 8 | 模块控制字节，bit7-bit1表示地址，  bit0表示读(1)或者写(0) |
| data\_in | input | 32 | 待写入EEPROM数据 |
| data\_out | output | 32 | 从EEPROM中读出数据 |
| req | input | 1 | 模块使能信号 |
| ack | output | 1 | 模块响应信号，1个时钟宽度的脉冲信号 |
| busy | output | 1 | 1. 模块忙标志位2. 也可用作读取EEPROM数据的数据有效标志位(低电平有效) |
| freq\_div | parameter | / | 控制SCL频率。计算公式为：  SCL频率 = 时钟频率/4/(freq\_div+1)  默认值124, 在50MHz时钟下，对应SCL频率为100KHz |
| cnt\_width | parameter | / | 分频计数器宽度，根据freq\_div选取合适的值即可，默认值为7 |

## 仿真验证

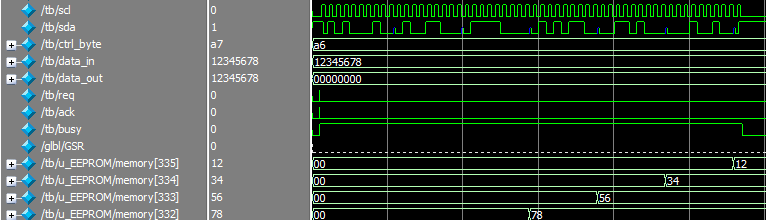
仿真借助EEPROM.v中的EEPROM model。但是由于此model不支持page write和 sequential read的读写操作模式，所以对该model做了一点修改。主要是在write\_to\_eeprm和read\_from\_eeprm两个task，使其支持page write和sequential read模式。特此说明。

1. 写EEPROM操作仿真。此次仿真希望将数据0x12345678写入地址0x14C-0x14F。

输入数据：

* data\_in = 0x12345678
* ctrl\_byte = a6 -> 地址为0x14C, 读写控制为0

将req拉高，通知模块开始进行EEPROM写操作。模块回复一个时钟周期的ack信号，同时busy信号置1，写操作开始。观察scl和sda信号，波形与预期一致。EEPROM model中的模拟momery的第332(0x14C), 333, 334, 335(0x14F)四个byte被写入指定数据，与预期一致。

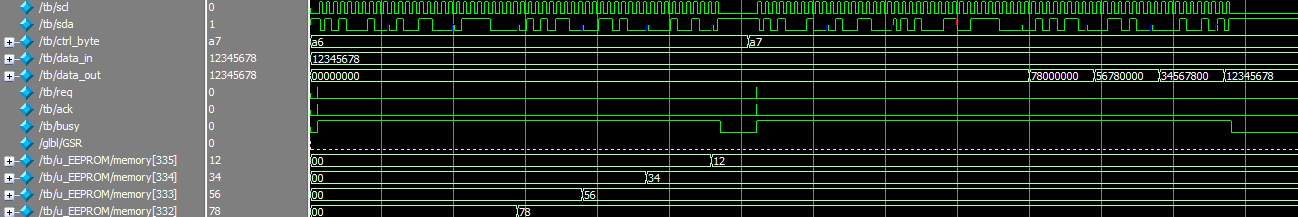


1. 读EEPROM操作仿真。此次仿真先进行一次写操作，将0x12345678写入EEPROM，然后进行读操作，希望将写入EEPROM的数据读出。关于前半段写操作的仿真就不在赘述。

读操作输入数据：

* ctrl\_byte = a7 –> 地址为0x14C, 读写控制位为1

将req拉高，通知模块开始进行EEPROM读操作。模块回复一个时钟周期的ack信号，同时busy信号置1，读操作开始。观察scl和sda信号，波形与预期一致。EEPROM model中的模拟momery的第332(0x14C), 333, 334, 335(0x14F)四个byte被读出到data\_out，读出数为0x12345678与预期一致。



## 板级验证

用之前开发的串口接收发送FIFO模块，从PC端发送数据。由于之前的串口模块是发送8byte，最后转换成的是64bit的数据。所以用前4个byte作为写入EEPROM的数据，第8byte作为模块的ctrl\_byte输入。同时用按键输入作为各个模块的使能信号（详见RTL）。

先在EERPOM的0x00-0x03地址写入0x01,0x02,0x03,0x04. ctrl\_byte = 0x00.

再在EEPROM的0x04-0x07位置写入0x02,0x03,0x04,0x05. ctrl\_byte = 0x02.

再在EEPROM的0x04-0x07位置写入0x03,0x04,0x05,0x06. ctrl\_byte = 0x04.

再在EEPROM的0x04-0x07位置写入0x04,0x05,0x06,0x07. ctrl\_byte = 0x06.

再在EEPROM的0x04-0x07位置写入0x14,0x15,0x16,0x17. ctrl\_byte = 0x08.

然后将之前写入的数据依次读出，ctrl\_byte依次为0x09,0x07,0x05,0x03,0x01.

FPGA将数据从EEPROM中读出后会通过串口发送回PC端，如下图所示，读出的数据（由于要补齐64bit数据，所以每一组数据都会多出4byte：EE DD CC AA）与写入数据相同。说明模块工作正常。

