I^2C 总线接口电路设计及 VLSI 实现

摘要： 本文提出了一个从模式的 I^2C 总线接口电路设计，该接口电路实现了对变参数 ASIC 器芯片的配置。此设计使得可配置的 ASIC 芯片中参数配置所需要的芯片管脚大大减少。该方案现以通过行为仿真、综合后门级时序仿真，并且在无锡上华 0.6um CMOS 工艺上实现。本文还提出了一个主从模式混合的 I^2C 总线接口控制电路设计，该接口电路即可以配置为主模式又可以配置为从模式，和微处理器配合可以很好的实现 I^2C 总线协议。该方案现以通过行为仿真、综合后门级时序仿真.

关键词： I^2C 总线接口电路、从模式、主从模式混合、 VHDL 、 VLSI

目 录

一、前言

I²C 总线是一种被广泛应用的芯片间的串行总线，该总线最早是由 Philips 公司提出并倡导的。 I²C 总线通过两条信号线实现了连接在总线上的器件之间的软件寻址和同步串行数据传输，完全避免了器件间的传统的片选寻址方法，节省了大量的数据总线、地址总线、控制总线所占用的芯片管脚和 PCB 板的面积。器件之间简单的 I²C 总线互联方式，可以使用户的硬件系统具有最简单而灵活的扩展方法、简单而有效的系统调试和故障排除方案。并且在设计基于 I²C 总线的用户系统时，由于功能模块图上的功能模块的互联可以直接对应实际的 I²C 总线器件的互联，因此可以很方便地把功能模块图转变成电原理图。如果再把常用的软件模块建库保存，在开发用户系统的软件部分所花费的时间和资源也可以大大减少。总之，基于 I²C 总线设计用户系统，可以有效降低用户系统的硬件部分和软件部分的研制成本和生产成本。图 1-1 为 I²C 总线的一个应用实例。

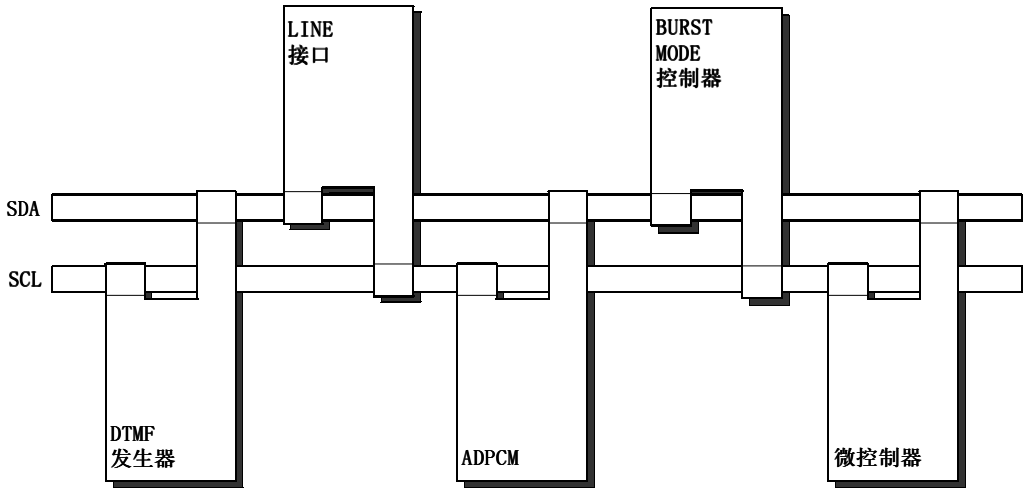


图 1-1　DECT 无绳电话基座

I²C 总线的发展简史：

- 1992 年 I²C 总线协议的 1.0 版推出。该版本保留了最高传输速率为 100kbt/s 的标准模式和 7 位寻址方式，并增加了最高传输速率达到 400kbit/s 的快速模式，具有该种模式的 I²C 总线接口的 IC 器件可以向下兼容，应用在 0-100kbit/s 的系统中。此外该版本还增加了对 10 位地址寻址方式的支持，这样就可以有 1024 个额外的从地址可以供系统使用。同时，该版本忽略了对从地址的软件可编程的支持以及低速模式，因为前者的实现过于复杂，未被实际应用，而后者事实上是整个 I²C 总线协议的一个子集，不需要另外加以描述。
- 1998 年 I²C 总线协议的 2.0 版推出。此时 I²C 总线协议已经成为了事实上的国际标准，被 IC 设计生产商广泛采用。该版本增加了高速模式，最高传输速率达到惊人的 3.4Mbit/s. 同时，原来对新的器件的固定输出电平的限制被取消，取而代之的是可以输出与总线电源

相关的电平；原来快速模式的输出级在 6mA 输出 0.6V 要求被忽略。

- 2000 年的 2.1 版开始允许在高速模式的重复启始信号后，可以延长 SCL 时钟的低电平周期，并放宽了高速模式的一些时间参数的要求。

本毕业设计中包含了一个由 VLSI 实现的从模式的 I^2C 总线接口电路和一个既可配置为主模式又可配置为从模式的 I^2C 总线接口电路。

前一个设计电路可作为一个成熟的 IP 核嵌入到参数可配置的 ASIC 芯片中，这样通过该接口电路只需两个管脚就可实现系统配置参数的输入和系统配置参数的输出验证,如图 1-2 所示。而完成相应功能的并行总线接口电路则需要相当多的管脚，比如一个 8 位总线接口至少需要 8 条数据总线、若干条地址总线、读写控制线和片选线。

后一个设计电路的功能更为强大，用途也更为广泛，不仅可以作为主模式 I^2C 总线器件配合从模式的 I^2C 总线接口电路完成对参数可配置的 ASIC 器件的系统配置参数的输入和参数的输出验证的任务，还可以直接和微处理器的 IP 核连接，使微处理器芯片直接具有 I^2C 总线接口，这样不需要再采用软件模拟的方式实现 I^2C 总线功能，因而大大节省了微处理器的硬件资源和运算资源,提高了微处理器的编程效率。

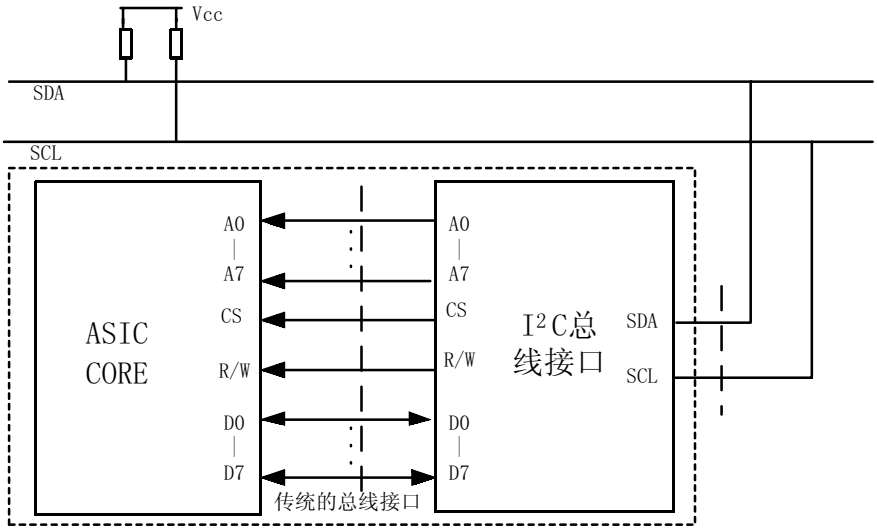


图 1-2 I^2C 总线接口示意图

二、 I²C 总线的基本原理

I²C 总线是通过两条双向线(时钟线 SCL ，数据线 SDA)在器件之间传递信息的。 SDA 、SCL 的输出级必须是开漏或者开集电极输出，并外接上拉电阻，以实现“线与”功能，如图 2-1 所示。当总线空闲时， SDA 、 SCL 均为高电平。总线上的每个器件都有一个唯一的地址，既作为数据的发送者或接收者，也可以配置成 master 或 slave 。 master 可以控制总线，开始数据传输，产生传输所需的 SCL 时钟脉冲。其他被寻址的器件只能处于 slave 模式。可以联入总线的器件数取决于总线的最大负载能力 400PF 。

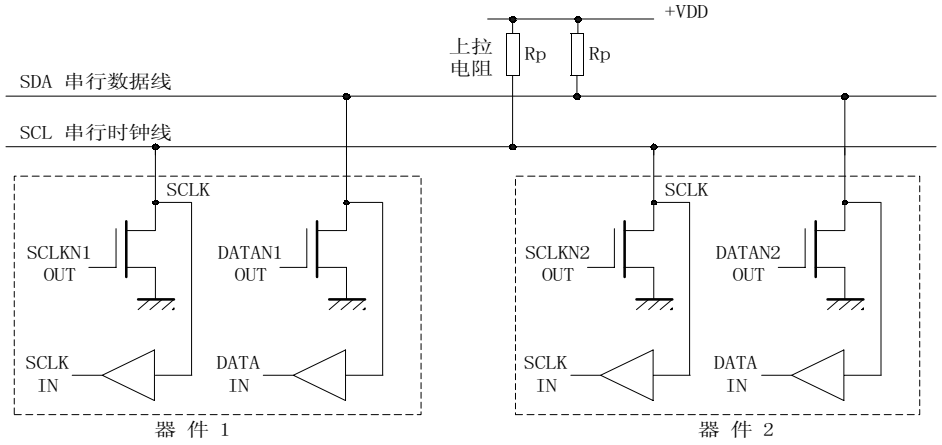


图 2-1 I²C 总线器件输出级示意图

I²C 总线上的数据传输由起始状态启动，由停止状态结束。 SDA 线上的数据在 SCL 时钟高时必须稳定。 SDA 上的高低电平状态的变化只能在 SCL 时钟为低时进行变化。在 SCL 为高时， SDA 由高到低的变化表示起始状态， SDA 由低到高的变化则表示停止状态。这样的定义可以避免把起始状态和停止状态误认作是普通的数据。

I²C 总线是一个允许有多个主控制器的总线，它定义了一个仲裁过程,用来保证当一个以上的 master 同时试图控制总线时，只有一个 master 可以控制总线，并且总线上传送的数据不被其他 master 破坏。由于 SDA 上的数据只有在 SCL 为高时有效，而所有的 master 在总线上传信息时都会在 SCL 上产生自己的时钟信号, 所以当有两个或两个以上的主控制器同时企图控制总线时,就需要用一个已经同步好的 SCL 时钟来进行以位为单位的仲裁过程。 SCL 总线时钟同步是利用了 I²C 总线接口的线与功能，只要一个器件的 SCL 输出低电平， SCL 时钟总线就会被成低电平。 SCL 时钟总线的拉低会使相关器件开始对自己的 SCL 低电平周期进行计数，只有当所有的器件对自己的 SCL 低电平周期计数结束回到 SCL 高电平周期时， SCL 总线才会被重新拉高。具有较短的 SCL 低电平周期的器件在计数结束后会释放自己的 SCL 输出同时进入等待状态，直到 SCL 总线被拉高。 SCL 总线拉高又会使相关器件对自己的 SCL 高电平周期计数，最先完成计数的器件会将 SCL 再次拉低。这样 SCL 的高电平周期是由拥有最短的 SCL 高电平周期的器件决定，而低电平周期则是由拥有最长的 SCL 低电平周期的器件决定。如图 2-2 所示。仲裁过程是当 SCL 时钟总线为高电平时，在

SDA 数据总线上进行的。当一个 master 的 SDA 输出为高，而另一个 master 的 SDA 输出为低时， SDA 数据总线线被拉低。这时前者发现 SDA 数据总线上的数据和自己的输出不一致，它意识到已经丧失了总线控制权，随即它将关闭自己的 SDA 输出，因此竞争胜利的 master 的数据不会被竞争失败的 master 破坏。如图 2-3 所示。失去总线控制权的 master 可以继续产生 SCL 时钟脉冲直到它失去总线控制权的字节传送完毕。

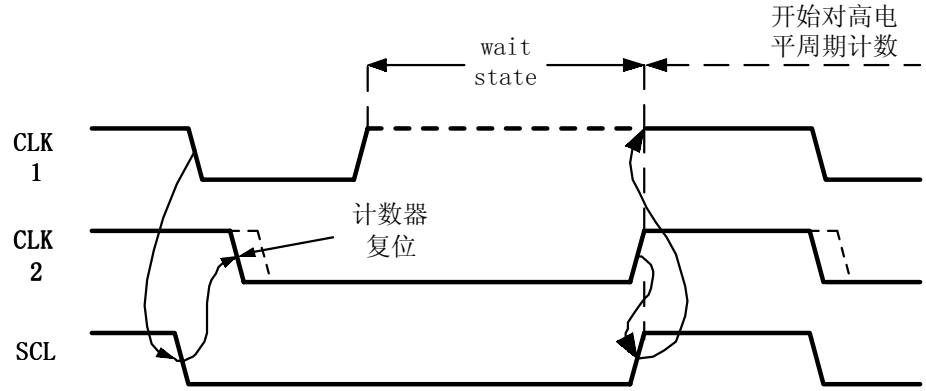


图 2-2 总线仲裁的时钟同步

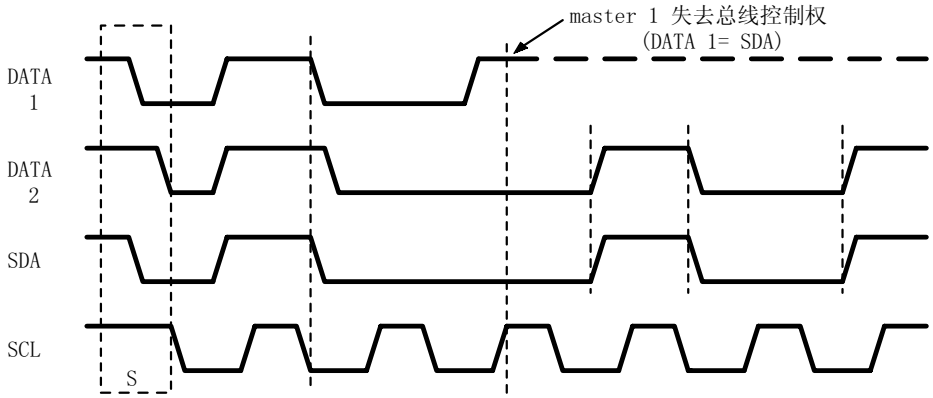


图 2-3 总线仲裁过程

I^2C 总线上的每个数据包都是由 8 个数据位和 1 个响应位构成，传送一个完整的数据字节就需要 9 个 SCL 时钟脉冲传送。发送方先发送数据的最高位，并在响应位释放 SDA ，接收方必须在响应位将 SDA 拉低表示接收到数据。 slave 接收者不能在响应位将 SDA 拉低，表示他不能接收数据，此时 master 可以产生停止信号中止传送。当 master 接收方不响应，表示这是接收的最后一个字节。

I^2C 总线的标准模式和快速模式的通讯是由 4 部分组成：起始信号、 slave 地址、数据传送、停止信号。本设计支持 7 位寻址方式。如图 2-4 所示。起始信号后， 7 位 slave 地址和 1 个读写控制位被发送。只有当 slave 的地址和 master 发送的地址一致时， slave 才会响应。 slave 被成功寻址之后，数据以字节为单位顺序传送，方向由读写控制位决定（“ 1 ”表示读，“ 0 ”表示写）， master 产生停止信号终止数据传输，释放总线。另外， master 也可以产生重复起始信号代替停止信号。

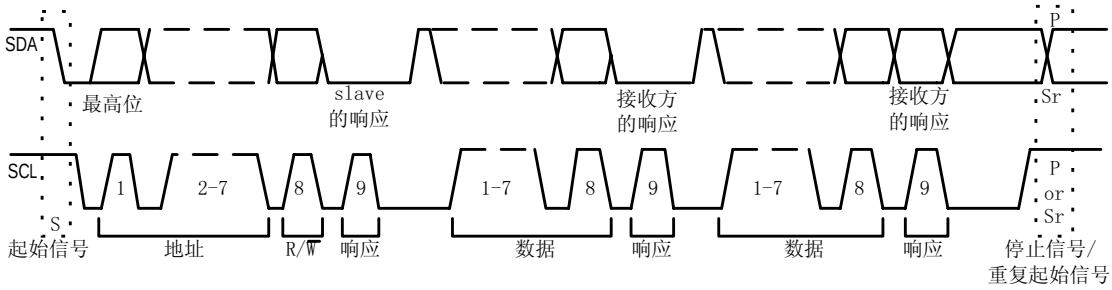


图 2-4　I²C 总线上的数据传输-7 位寻址方式

I²C 总线协议还支持 10 位寻址方式。该寻址方式利用在 7 位寻址方式中保留的地址(11110XX)作为 10 位寻址的第一个字节，这样可以保证支持 7 位和 10 位寻址方式的器件可以存在于同一个系统中。启始信号后， master 发送 10 位寻址的第一个字节(11110XX+读写控制位， XX 代表 slave 地址的最高两位；读写控制位为 0，则表示主控器向从控制器写信息,第二个字节将包含寻址地址的最后 8 位，为 1 表示主控器向从控制器读数据,第二个字节包括由从控制器返回的数据)。支持 10 位寻址的 slave 器件收到该寻址字节，就与自己的 10 位地址的最高两位进行比较,如果一致， slave 就响应 master 。当读写控制位为‘0’时，所有响应的 slave 将继续接收并比较由 master 发送的寻址的第二个字节，如果地址再次符合， slave 再次响应 master ，这时 master 向 slave 写的寻址过程已经完成。接下来可以进行 slave 接收 master 发送的数据字节的工作，或者 master 可以发送重复启始信号，然后再发送读写控制位变为‘1’的 10 位寻址的第一个字节，刚才被寻址的 slave 将响应 master ，此时完成了 aster 向 slave 读的寻址过程。接下来 master 可以被寻址的 slave 读取数据。 10 位寻址方式过程如图 2-5 所示

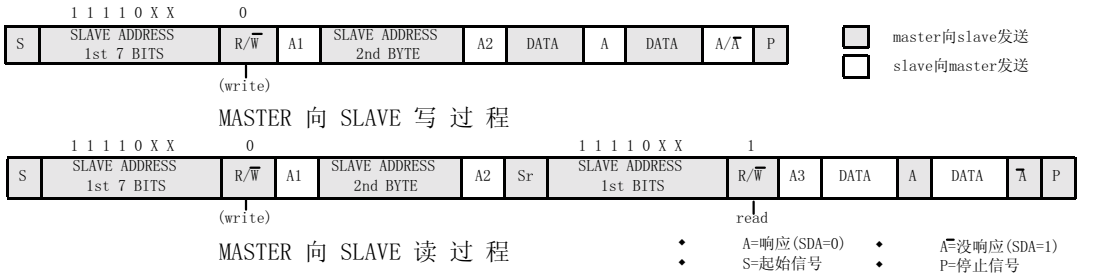


图 2-5 I²C 总线 10 位寻址方式

由于该设计中的从模式的 I²C 总线接口电路是面向可配置参数的 ASIC 芯片设计配置所需要的 I²C 总线接口电路,要求通过软件寻址不仅访问芯片本身，而且还要能访问芯片内部寄存器组阵列。因此， master 和 slave 在总线上的通讯过程定义如图 2-6 所示：

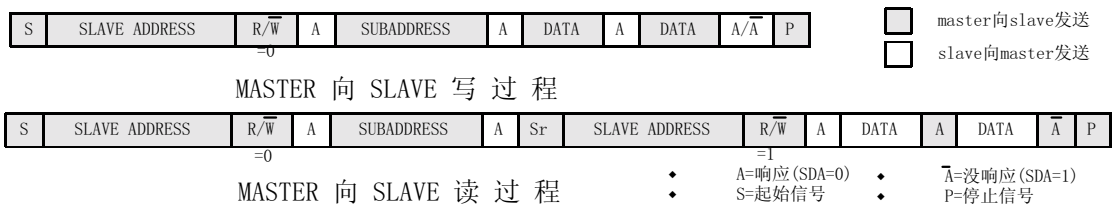


图 2-6 I²C 总线通讯过程的定义

I²C 总线所支持的高速模式向下兼容快速模式和标准模式,其速度最高可达 3.4Mbit/s,高速模式除了没有仲裁和时钟同步以外,其总线协议、数据格式和快速模式相同。为了达 3.4Mbit/s 的速度，高速模式的器件在 sda 输出端有一个开漏极缓冲器，在 SCL 端有开漏极下拉电路和电流源上拉电路。电流源电路是为了减少 SCL 的上升时间，并且只有在高速模式下，该电路才会起作用。为了加快数据处理速度，每一个 master 都分配了一个特定的器件标号，总线仲裁的过程是在快速模式下传输 master 器件标号时完成的。

高速模式在出现下列条件后才开始

1.起始信号　　2.8 位 master 器件标号　　3.非应答位

在非应答位后,SCL 回到高电平,master 将转成高速模式并开启电流源上拉电路.如果 slave 要延迟数据传输,就必须在特定时刻之前延长 SCL 信号的低电平周期,当 slave 释放 SCL 以后,master 开启电流源上拉电路,减少 SCL 信号的上升时间。在每一个重复起始信号和一个应答或非应答信号之后， master 将关闭电流源上拉电路， slave 这时可以延长 SCL 信号的低电平周期从而延迟数据传输。当 slave 都释放了 SCL 以后，主控器将开启电流源上拉电路，从而减少 SCL 信号的最后部分的上升时间。高速模式只有在出现停止信号后才被关闭。

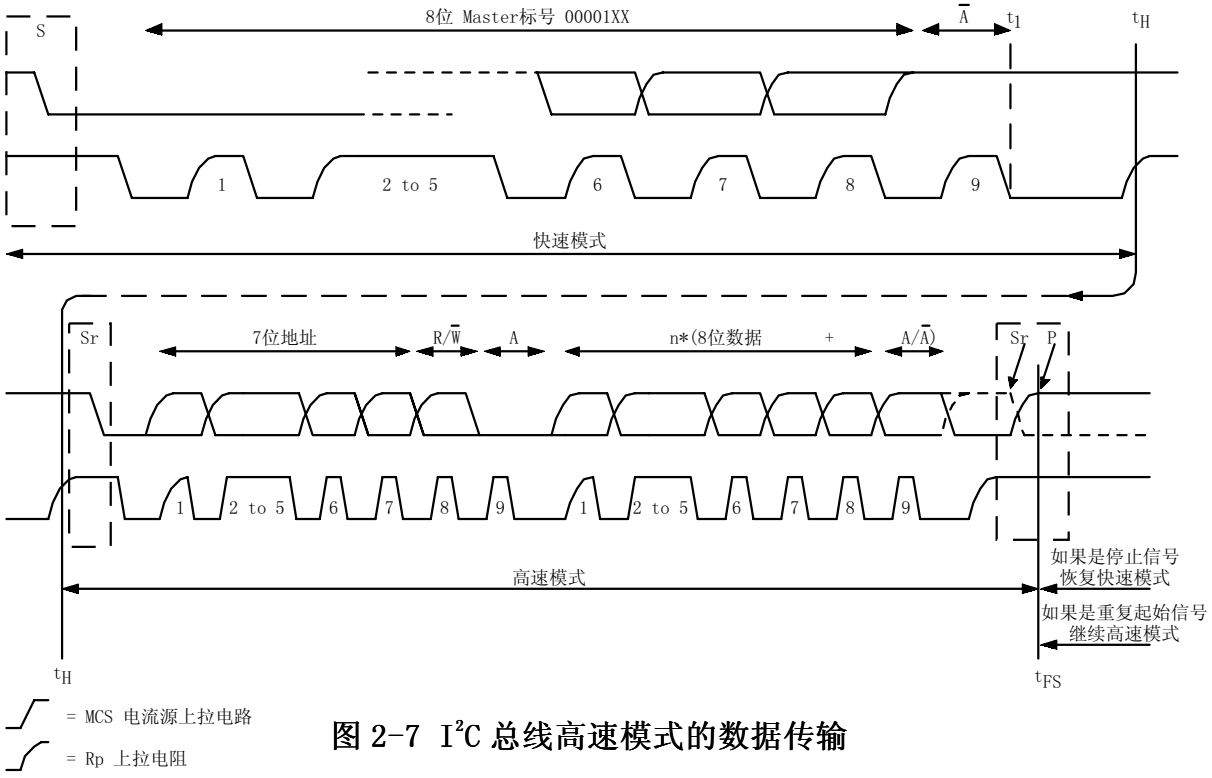


图 2-7 I²C 总线高速模式的数据传输

三、从模式的 I²C 总线接口的电路结构设计

3.1 系统结构

该接口电路主要用于芯片内部寄存器的读写访问，因此只需要实现从模式即可满足要求，这样可以避免占用过多的芯片面积资源。

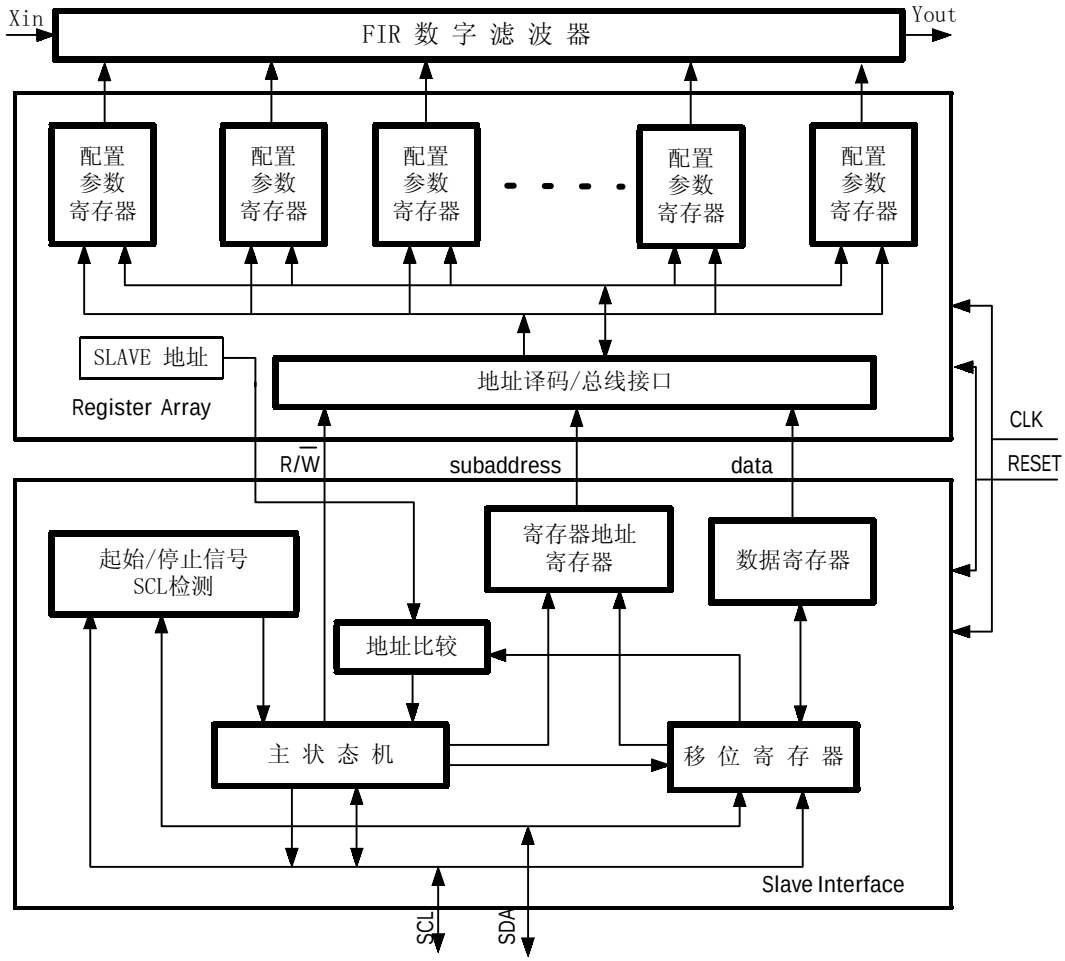


图 3-1 系统框图

根据 I²C 总线上数据传输过程的特点，使用状态机控制整个系统的运作，可以使设计思路清晰明确，程序模块化。系统上电复位后即进入状态机，并在状态机的控制之下，完成起始/停止信号的自动检测， slave 地址、子地址、配置数据的接收及响应，发送数据及检测响应，对配置寄存器阵列的寻址和连续读写（子地址自动增加）等功能。此外，系统最高数据传输速度应由系统时钟 clk 及寄存器的存取速度决定，以适应 100KHZ 和 400KHZ 两种模式。根据电路的功能，整个系统可分为几部分，如图 3-1 所示。

3.2 主状态机

各状态说明

Idle	空闲状态		
rx_address	接收 slave 地址	rx_subaddress	接收子地址
ack_rx_address	接收 slave 地址响应	ack_rx_subaddr	接收子地址响应
tx_data	发送数据	rx_data	接收数据
ack_tx_data	发送数据响应	ack_rx_data	接收数据响应

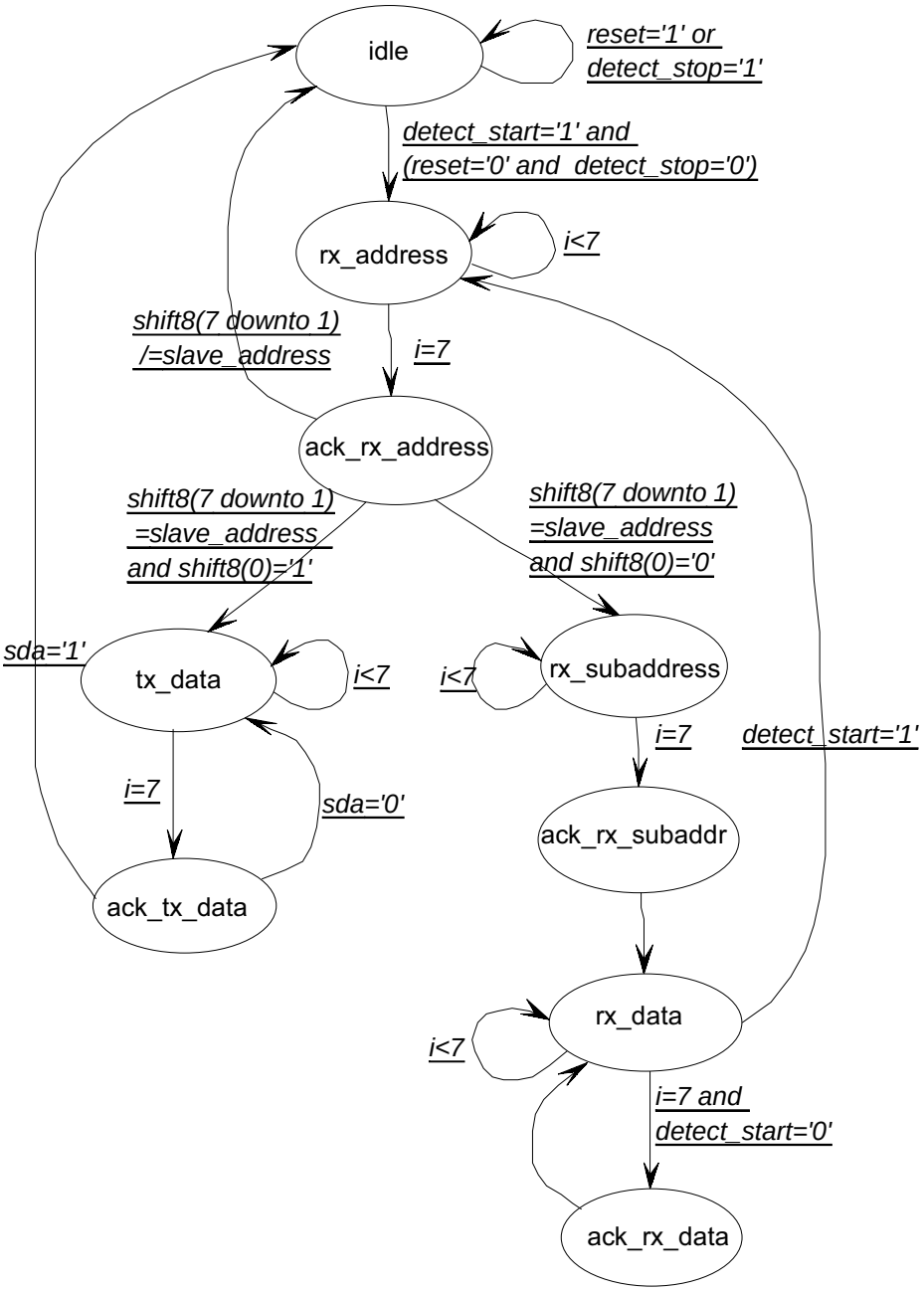


图 3-2 主状态机

状态机的状态转换是在SCL下降沿后第4个时钟周期进行的。但是，停止信号出现后总线被释放， SCL 、 SDA均为高电平，除非再出现起始信号进行数据传送， SCL是不会再出现下降沿的。如果停止信号出现后就把空闲状态赋给next_state ，将没有SCL时钟下降沿做时间基准完成状态转化(即状态机进入到空闲状态)。为了解决矛盾，需要将停止信号看作复位信号。一旦有停止信号出现，当前状态在下一个CLK脉冲强制进入到空闲状态,而不能等到SCL下降沿后第4个时钟脉冲。

ack_rx_subaddr状态之后， master既可能发送数据又可能发送重复停止信号或者停止信号，如何区分是哪种情况成为正确进入相应的状态的关键。最好的办法是ack_rx_subaddr状态之后无条件紧跟rx_data状态。这样，如果在rx_data状态时检测到重复起始信号，丢弃已接收的数据，并在SCL时钟下降沿后第4个时钟周期进行状态转化进入rx_address状态，，否则继续完成接收数据。若发现停止信号，处理如前所述。这样做既满足设计要求又不会增加多余的状态。

master收到数据后没有响应(SDA=1) 表示这是master接收的最后一个字节数据。在这种情况下，可以在SCL时钟下降沿后第4个时钟周期就直接进入到空闲状态，处理接下来应该出现的重复起始信号或停止信号。

3.3 子模块

3.3.1 起始/停止信号、 SCL检测

在高速时钟的上升沿采样SDA 、 SCL，如果在前后两个采样点发现SDA从“1”变化到“0”而SCL保持“1”则认为master发出起始信号，将detect_start置为有效。同理,若发现SDA从“0”变化到“1”而SCL保持“1”,则认为是停止信号,将detect_stop置为有效。 Detect_start有效保持一个状态周期,detect_stop有效保持一个时钟周期,以避免误操作。

依靠SCL时钟下降沿进行状态转换以及在SCL时钟上升沿将SDA的数据送入移位寄存器，时间有些仓促，因此决定推迟若干个 c1k 时钟周期进行。最简单的实现办法是将 SCL延时若干个时钟周期，再采样检测延时后的 SCL 的下降沿和上升沿分别作为状态转换和数据采样时刻。

3.3.2 移位寄存器

移位寄存器在rx_address 、 rx_subaddress、 rx_data状态完成在数据采样时刻(SDA数据有效时)对slave地址、寄存器子地址、数据的移位接收，接收完毕在响应状态将数据并行送往指定的寄存器；在 ack_tx_data 状态和 ack_rx_address 状态且读写控制位为高(r/w=1)时,并行输入数据，在 tx_data 状态发送时刻有效时输出最高位，在数据采样时刻有效时进行移位操作。

3.3.3 寄存器子地址寄存器、计数器寄存器

寄存器子地址寄存器实现完成对寄存器一次读写操作后地址自动增加的功能 ，这样

可以实现在对器件的配置时只输入一次子地址就完成对所有的配置寄存器的配置。计数器寄存器记录接收或发送的数据的位数，为状态机提供控制信号。

3.3.4 SDA 、 SCL 双向线

利用三态门电路实现开漏或开集电极输出电路。如图 3-3 所示

电路实现：

```
entity bidirec is
   port (SDAout:in std_logic;
    SDAen:in std_logic;
    SDAin:out std_logic;
    SDA:inout std_logic);
end bidirec;
architecture bidir of bidirec is
begin
process (a,e)
begin
    case SDAen is
        when '1' => y<=SADout;
        when '0' => y<='Z';
        when others =>y<='X';
    end case;
 end process;
    SDAout<=SDA;
end bidir;
```

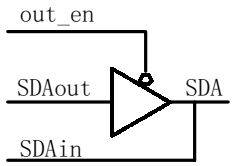


图 3-3 双向线示意图

为了减少外加干扰对接口电路的不良影响，因此只在 SCL 为高并且在特定的状态时才允许 SDA 输入。为此增加 SDA 输入使能信号，该信号由状态机决定。

3.3.5 RAM 单元

本设计在从模式 I^2C 总线接口电路后接 RAM 单元，使之成为一个可实用具有的 I^2C 总线接口电路的静态 RAM 电路。该 RAM 是由 D 触发器组构成的.由从模式的 I^2C 总线接口电路得到的 RAM 地址经过译码选中 D 触发器组，在读写线的控制下对该 D 触发器组进行读写操作。

四、主从模式混合的 I²C 总线接口的电路结构设计

4.1 系统结构

4.1.1 设计目标

设计一个功能完善的 I²C 总线接口电路，该接口电路可以被配置为主模式或从模式的接收方和发送方，主从模式之间可以相互切换，特别是当总线控制权失去后，该电路可以立刻由主模式转成从模式，不需要额外的指令。该电路可自动检测和产生启始信号和停止信号，在主模式下进行总线仲裁过程和总线空闲状态的检测。该电路不仅应该具有 I²C 总线的接口， 而且还应该具有微处理器的并行接口,便于和微处理器直接相联，传递数据，交换控制信息和状态信息。

4.1.2 系统框图

根据设计目标，将整个系统划分为 I²C 总线控制器电路和微处理接口电路两大部分，如图 4-1 所示。

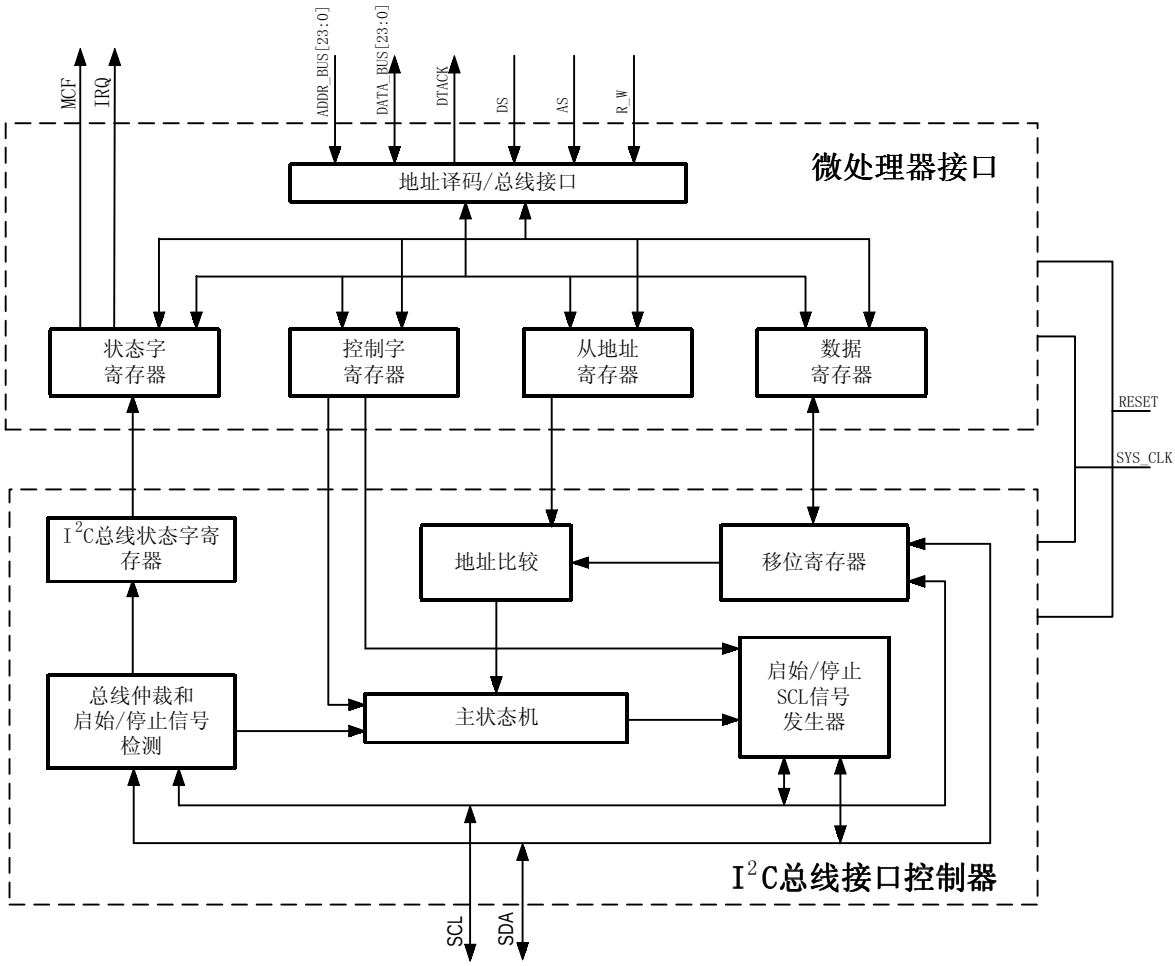


图 4-1 系统框图

4.2 I^2C 总线接口控制器子模块

4.2.1 总体设计思路

在设计支持主从模式混合的 I^2C 总线接口电路时，有两种设计思路可供选择。

第一种，用两个相对独立的两个状态机分别描述系统的主模式和从模式的运作。这两个状态机同时运行，分别产生自己的 SCL 、 SDA 输出。它们的 SCL 、 SDA 输出信号经过主从模式选择信号的选择后输出。主从模式选择信号是由外接的处理器决定的，当控制器失去总线控制权之后，主从模式选择信号可以自动由主模式转为从模式而不另外需要外接处理器的命令。这样就可以保证该接口控制器既可以被配置为主模式又可以被配置为从模式，而且当控制器在主模式的寻址阶段失去总线控制权时，能够立即转入从模式并且被其他竞争胜利的 master 寻址。

第二种思路也采用双状态机进行描述。所不同的是第一个状态机虽然也负责产生主模式下的 SCL 、 SDA 输出，但其 SDA 的输出除了在总线空闲状态、产生启始信号和停止信号时由本状态机决定以外，其余过程的 SDA 主模式输出将由另外一个状态机即主状态机决定。主状态机将主模式和从模式的系统过程比如 SDA 信号输入输出过程合并在了同一个状态机中，这是因为主模式和从模式下的寻址或被寻址过程及数据传送过程所使用的状态和操作几乎完全相同。该状态机是由 SCL 时钟信号进行驱动，完成状态转换的。 SCL 时钟信号既可以是第一个状态机产生的 SCL 时钟信号(主模式下)又可以是另外的 master 产生的 SCL 时钟信号(从模式下).这样的设计实际上是将整个设计的行为级进行了分层，下层产生的信号驱动上层状态机的运作，而上层不管驱动信号的来源，这样上层的驱动信号的产生不需要只限于同一个器件。这样的分层设计也同样可以保证实现所有的目标功能，因为主从模式也同样是在同时运行的，其输出的 SDA 信号经过主从模式选择信号的选择进行输出。

这两种设计思路是从不同的角度对系统进行行为级描述的，前者是横向上将主从模式进行简单的组合，后者则在纵向的两个层次上对系统进行描述的。相比较而言，后者的设计思路相对更为先进一些，这样的设计思路有些和计算机网络的分层协议类似，思路清晰，每一层只需要完成特定的功能，易于实现，而且只要层次间的接口明确，就可以用其他的电路实现模块来替代原来的模块,这样功能易于扩展。因此如果采用后一种设计思路，在进行主从模式混合设计时，就不需要考虑到底层 SCL 信号的来源和具体产生过程，这样很容易在高层将主从模式混合，达到比较满意的效果。本设计即采用后者的设计思路。

4.2.2 状态机

4.2.2.1 SCL 状态机

该状态机产生主模式下的 SDA 、 SCL 输出和起始、停止信号。

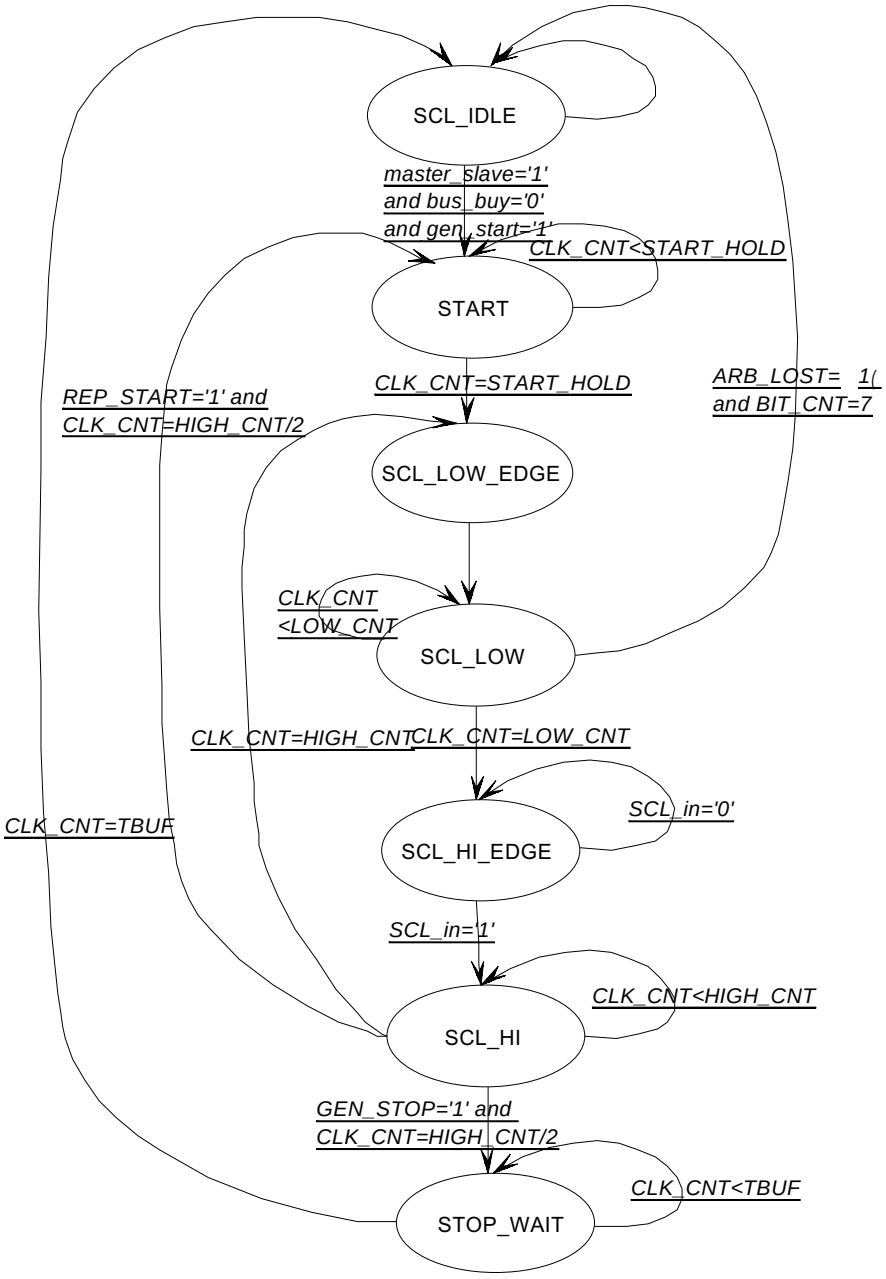


图 4-2 SCL 状态机

在 SCL_IDLE 状态,SDA 、 SCL 输出高阻，此时其他的 master 可以控制总线。一旦需要产生启始状态且控制器处于 master 状态及总线空闲，状态机就会进入到 START 状态。

在 START 状态， SCL 输出高阻、 SDA 输出低电平产生起始信号，同时系统时钟开始计数。当 START 状态保持时间满足后，状态机进入 SCL_LOW_EDGE 状态。

在 SCL_LOW_EDGE 状态产生 SCL 的下降沿， SDA 则保持原来的输出状态,系统时钟复位。状态机随即进入到 SCL_LOW 状态。

在 SCL_LOW 状态,SCL 输出为低电平，系统时钟开始计数。数据保持时间满足后，如果 RE_START 有效， SDA 输出高阻，为产生重复启始信号做准备；如果 GEN_STOP 有效， SDA 则被拉低，为产生停止信号做准备，否则 SDA 根据 master_sda 决定输出。当 SCL_LOW 保持时间达到时，如果控制器失去了总线控制权而且在失去控制权的那个字节传送完毕后，状

态机回到 SCL_IDLE，停止产生 SCL 时钟，否则状态机将进入到 SCL_HI_EDGE 状态。

在 SCL_HI_EDGE 状态,系统时钟复位， SCL 输出高阻， SDA 保持原输出状态。如果采样到 SCL 时钟总线仍然为低，状态机将保持 SCL_HI_EDGE 状态，等待 SCL 总线被其它的控制器释放，用以满足 SCL 时钟同步协议。当采样到 SCL 时钟变成低电平时，状态机进入到 SCL_HIGH 状态。

在 SCL_HIGH 状态， SCL 输出高阻， SDA 保持原输出状态,系统时钟开始计数。如果需要产生重复起始信号，当达到 SCL_HIGH 保持时间的一半后，状态机转入 START 状态；如果需要产生 STOP 信号，达到 SCL_HIGH 保持时间的一半后，状态机进入 STOP_WAIT 状态；否则，状态机在 SCL_HIGH 保持时间满足后回到 SCL_LOW_EDGE 状态。

在 STOP_WAIT 状态，系统时间继续计数， SCL 输出高阻， SDA 输出高阻，产生停止信号，并释放总线。这个状态是为了保证控制器在连续产生停止信号和起始信号之间有足够的时间间隔而设立的。当保持时间满足后，状态机回到 SCL_IDLE 状态。

4.2.2.2 主状态机

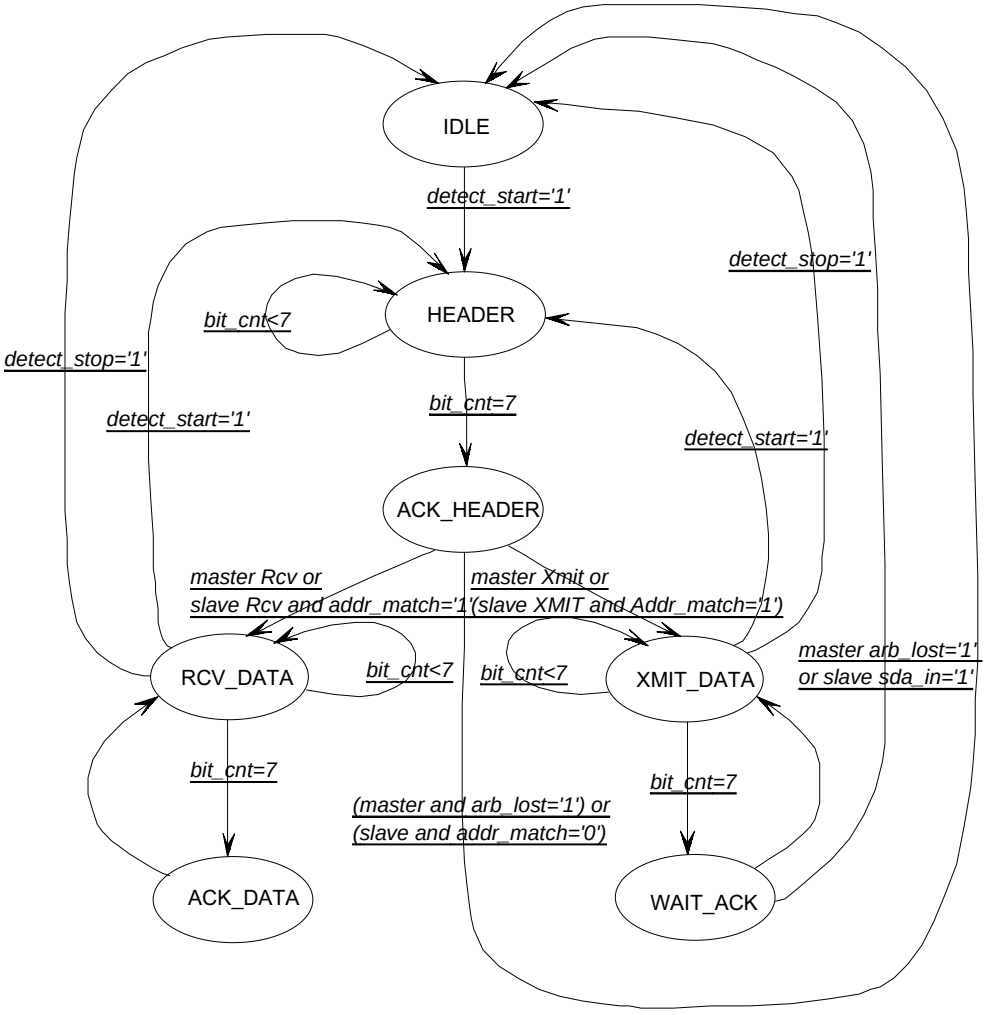


图 4-3 主状态机

该状态机融合了主模式和从模式两种模式，该状态机在系统时钟控制下，在detect_falling_edge 有效(指示 SCL 的下降沿)时完成状态转换.SCL 时钟可以是由任何master 产生。在每个状态，系统检查系统处于何种模式以决定正确的系统输出和进入下一个状态的条件。这样就可以实现当主模式失去总线控制权后快速转成从模式或者该接口可以被其他 master 寻址。

系统上电复位后，状态机进入到 IDLE 状态。当系统检测到总线上出现启始状态即detect_start 有效时，系统进入 HEADER 状态。

在 HEADER 状态，状态机传送寻址字节。从模式串行接收总线上传来的 slave_address，而主模式，则一面串行输出 slave_address ，一方面同时接收总线上传送的slave_address 。这样，可以保证当主模式失去总线控制权转入从模式时，系统可以被其它master 寻址。 slave_address 传送结束，状态机进入 ACK_HEADER 状态。

在 ACK_HEADER 状态：从模式，如果系统被寻址，则系统响应 master 并根据读写控制位进入到 RCV_DATA 或者 XMIT_DATA 状态，否则系统不响应并返回 IDLE 状态。主模式，则根据读写控制位直接进入 RCV_DATA 或者 XMIT_DATA 状态,如果 slave 不响应，就需要 SCL状态机在下一个状态产生停止信号。

在 RCV_DATA 和 XMIT_DATA 状态，移位寄存器串行接收或者发送数据,如果检测到启始信号则状态机进入 HEADER 状态；如果检测到停止信号，状态机会被强制进入 IDLE 状态，这时是不能在 detect_falling_edge 有效完成状态转化的,因为停止信号后是不会出现 SCL的下降沿的，除非再次出现起始信号，这就需要提升对停止信号响应的优先权，一旦发现停止信号，状态机就会在系统时钟的上升沿进入到 IDLE 状态，从而完成状态转换；如果主模式失去了总线控制权，状态机使 SDA 输出高阻，释放总线的控制权。 当接收或者发送完毕，状态机分别进入 ACK_DATA 或者 WAIT_ACK 状态。

在 ACK_DATA 状态，在主模式下系统如果希望从 slave 处继续接收数据，就需要响应slave ，否则不响应，说明这是 master 接收的最后一个字节。从模式，一般情况下都响应master 的。状态机回到 RCV_DATA 状态。

在 WAIT_ACK 状态，在从模式下系统发现 master 没有响应，状态机需进入到 IDLE 状态释放总线，以使 master 产生停止信号或者重复启始信号；否则，状态机回到 XMIT_DATA 状态。在主模式下，如果系统发现 slave 响应，则状态机进入 XMIT_DATA 状态；否则，状态机返回 IDLE 状态，同时需要 SCL 状态机在 XMIT_DATA 状态时产生停止信号终止数据传送。

4.2.3 各子模块

4.2.3.1 仲裁过程子模块

该模块检查系统在主模式下是否总线竞争失败。在 IDLE 、 HEADER 和 XMI_DATA 和ACK_DATA 状态且 SCL 时钟为高电平时，主模式下的该系统检查 SDA 总线上的数据和系统内部的 SDA 输出是否一致，如果不同则说明系统总线竞争是否失败， arb_lost 置为有效，同

时系统由主模式转成从模式，释放总线，主从模式选择信号复位。这样就可以保证在正确的时间点使 arb_lost 有效，并可以覆盖发生总线竞争失败的各种情况。

总线竞争失败发生在下列几种情况：

- 在放送数据周期和接收数据周期的响应位 SDA 为低而系统内部的 SDA 输出为高
- 在总线忙时，试图产生启始信号
- 从模式下，试图产生启始信号
- 主模式没有要求产生停止信号时，检测到了停止信号

如果系统在数据传输过程中失去了总线控制权， SCL 状态机将继续产生 SCL 时钟信号直到失去总线控制权时正在传送的那个字节传送完成。

4.2.3.2 信号输入 信号检测子模块

该模块负责监测 I^2C 总线的状态。系统高速采样 SDA 、 SCL 输入信号，如果在前后两个采样点发现 SDA 从“ 1 ”变化到“ 0 ”而 SCL 保持“ 1 ”就认为总线上出现启始信号，detect_start 置为有效,同时 bus_busy 也置为有效。同理,若发现 SDA 从“ 0 ”变化到“ 1 ”而 SCL 保持“ 1 ”,则认为出现停止信号,将 detect_stop 置为有效,bus_busy 置为无效。Detect_start 有效只保持一个状态周期而 detect_stop 有效只保持一个时钟周期，以避免误操作。如果发现 SCL 在前后两个采样点发生跳变(即 SCL 的上升或和下降沿)，则置 detect_rising_edge 或者 detect_falling_edge 有效，有效也只保持一个系统时钟周期。

4.2.3.3 控制字子模块

该模块高速采样 MSTA 输入信号 (主从模式选择信号，‘0’为从模式，‘1’为主模式)，当检测到 MSTA 的上升沿，置 gen_start 有效，指示 SCL 状态机应该产生启始信号；当检测到 MSTA 的下降沿且控制器没有失去总线的控制权，置 gen_stop 有效，则说明应该产生停止信号； gen_start 和 gen_start 分别在检测到启始信号和停止信号后失效。当控制器失去总线控制权， MSTA 则自动置 0 。系统处于何种模式是由 master_slave 决定，‘1’表示主模式，‘0’表示从模式，该信号由 MSTA 决定。

4.2.3.4 状态字子模块

该模块产生指示系统状态的输出信号，以便 I^2C 控制器和外接的 MCU 进行握手过程。例如 MCF(指示数据传送完成)， MASS(指示系统被其他的 master 寻址)， MBB(指示总线是否占用)， MAL(指示系统是否竞争总线失败)， RXAK(收到的响应位)等。

4.2.3.5 移位寄存器子模块

该移位寄存器负责数据的串并转换。在 HEADER 、 RCV_DATA 、 XMIT_DATA 状态，当 detect_rising_edge 有效时 (即 SCL 的上升沿)该寄存器进行移位操作，输入 SDA 上的数据，并在数据发送时刻输出最高位。在 IDLE 状态且 detect_start 有效时、 ACK_HEADER 状态且下一个状态为 XMIT_DATA 时或者 WAIT_ACK 状态时，该寄存器装入需要发送的数据等待发送，在 ACK_DATA 状态并行输出已经接收的数据。

4.2.3.6 时钟计数子模块

该模块为 SCL 状态机提供时间基准。当计数使能信号有效且清零信号无效时，计数器对系统时钟脉冲信号进行计数(或者叫做计时),否则该计数器将被同步清零。计数使能信号和清零信号均是由 SCL 状态机控制产生的。

4.2.3.7 传送数据位数子模块

该模块在发送或接收完一位数据后， SCL 时钟下降沿 bit_cnt 计数器自动加 1,在 IDLE 或者响应状态，该计数器清零。

4.2.3.8 双向接口子模块

利用三态门电路实现开漏或开集电极输出电路

4.3 微处理器接口模块

该接口模块把微处理器的并行数据总线上的数据送到系统内部的寄存器中，并把 I^2C 总线控制器配置为主模式或者从模式进行发送或接收数据的操作。

微处理器向 I^2C 控制器发送数据的握手过程：

第一步、 MCU 设置读写控制线， MCU 把地址放到地址总线,置地址指示脉冲(AS)为低，MCU 把数据放到数据总线上，置数据指示脉冲(DS)为低； I^2C 控制器在 AS 为低的时候进行地址译码，在 DS 为低时锁存数据，同时置响应信号 Dtack 为低。

第二步、 MCU 检测到 Dtack 为低， DS 、 AS 均被置高，数据线上的数据被清除。

第三步、 I^2C 控制器 Dtack 置为高电平。数据传送周期结束。

I^2C 控制器向微处理器发送数据的握手过程：

第一步、 MCU 设置读写控制线， MCU 把地址放到地址总线,置地址指示脉冲(AS)为低，MCU 置数据指示脉冲(DS)为低； I^2C 控制器在 AS 为低的时候进行地址译码，在 DS 为低时将数据发到数据线上，同时置响应信号 Dtack 为低。

第二步、 MCU 将数据线上的数据锁存， DS 、 AS 均被置高。

第三步、 I^2C 控制器清除数据线上的数据， Dtack 置为高电平。数据传送周期结束。

4.3.1 微处理器接口内部的状态机

该状态机是由系统时钟的上升沿驱动完成状态转化的。

系统上电复位进入到 IDLE 状态。在这个状态，微处理器可以把地址放到地址总线上，读写控制线置为正确的状态，置地址指示脉冲信号(AS)和数据指示脉冲信号(DS)为低电平。地址指示脉冲信号和数据指示脉冲分别指示地址总线上的地址和数据总线上的数据有效。如果是写周期，微处理器将数据放到数据线上。如果状态机检测到 AS 的下降沿，状态机进入到 ADDR 状态，否则继续保持 IDLE 状态。

在 ADDR 状态，如果系统被微处理器寻址(即 address_match 有效)且数据线上的数据有效(即 DS 为低) 时，系统进入到 DATA_TRS 状态进行数据传送。否则，状态机因为本器件因为没有微处理器被寻址或者数据线上的数据无效而回到 IDLE 状态。

在 DATA_TRS 状态，如果是写周期，系统将数据线上的数据锁存到相应的寄存器中；如果是读周期，系统将数据放到数据线上；状态机随后进入到 ASSERT_DTACK 状态，向微处理器响应(即置 Dtack 为低电平)，说明数据准备好(读周期)或者数据已经收到(写周期)。微处理器检测到 Dtack 为低电平，如果是写周期就将数据总线上的数据清除；如果是读周期，将数据总线上的数据锁存；并将读写控制线设为读状态， AS 、 DS 置为高电平。当系统检测到 AS 被拉高，状态机返回 IDLE 状态，等待下一个数据传送周期的到来。

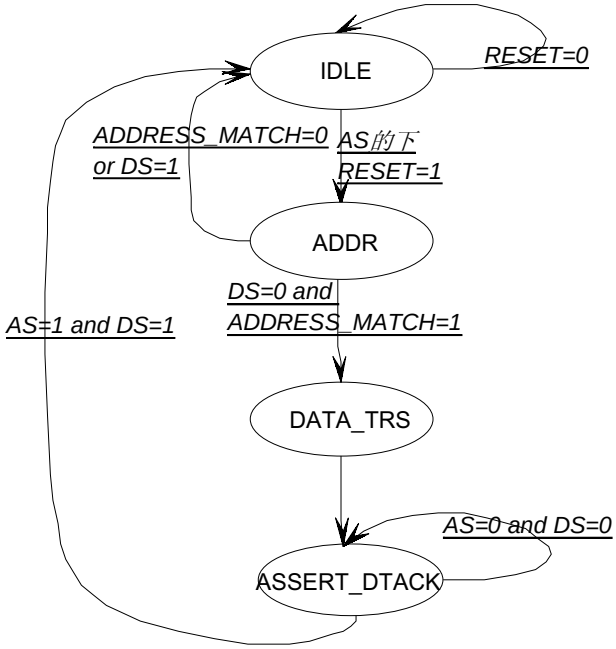


图 4-4 微处理器接口内部的状态机

4.3.2 地址译码

微处理器接口模块内部有 4 个寄存器器-数据寄存器、地址寄存器、控制寄存器和状态寄存器-每个寄存器都有自己的地址，可以被微处理器寻址访问。这样的设计可以将原来需要占用芯片管脚的各寄存器的输入输出线合并，共用数据线进行输入输出操作，有效的减少了对芯片管脚的占用。同时，这样也需要将地址总线上的寄存器地址进行译码，以访问这几个寄存器。地址译码采用 case-when 语句描述，当地址总线上的地址和寄存器的地址一致时，寄存器的使能信号有效，就可以在读写控制线的控制下对该寄存器进行读写操作。

4.3.3 读写操作

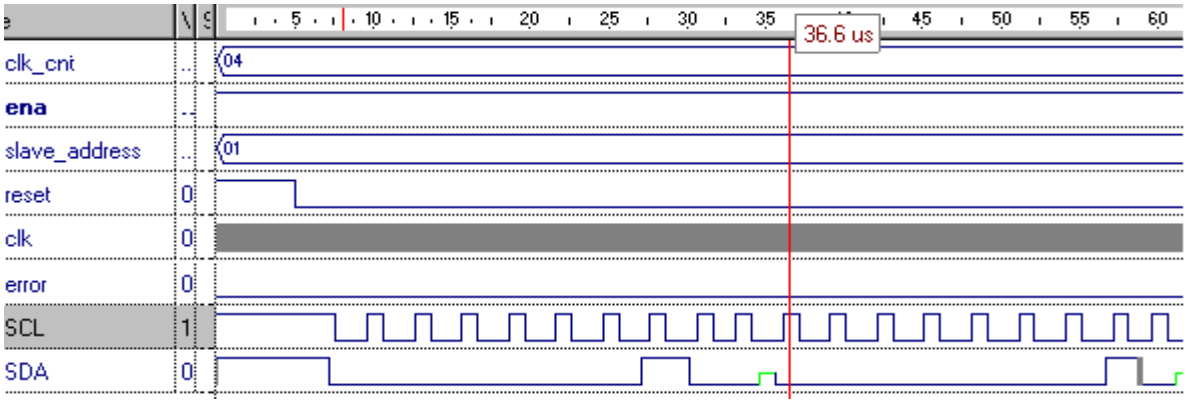
在时钟信号的控制下，再根据寄存器使的能信号是否有效、寄存器的读写权限的限制和读写控制线的状态，对给寄存器进行读写操作。

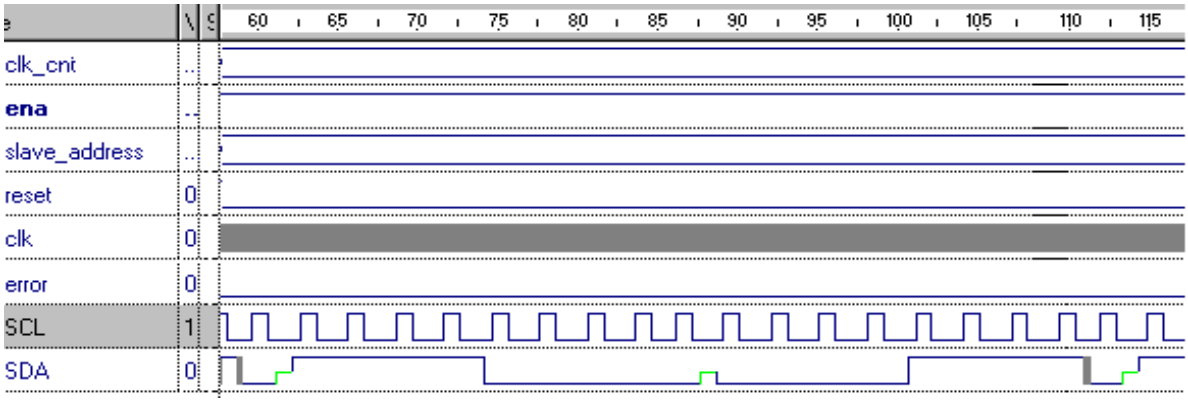
五、测试验证

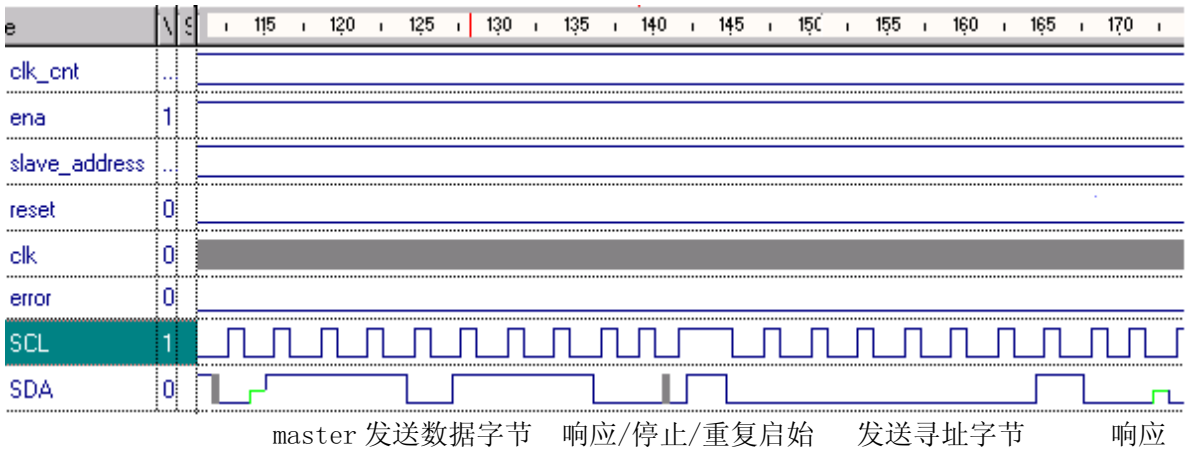
5.1 从模式的 I²C 总线接口电路的测试

为了验证该接口电路的功能，比较简单的方法是用 master 直接向该 I²C 总线接口电路进行数据传送。由于需要验证的 I²C 总线接口电路内部有 ram 单元，可以保存数据，因此只要 master 可以从 slave 那里读回先前发送的数据，即可以说明该接口电路功能正确。

master 电路的设计是将主模式下的通讯过程划分为若干个不同的小过程，每个小过程完成一个特定的任务，比如产生启始信号、发送一位数据等等，并产生自己的 SCL 、 SDA 输出，再把这些小过程通过上层命令调用的方式进行有机的联接即可。模拟结果，如图 5-1 所示：







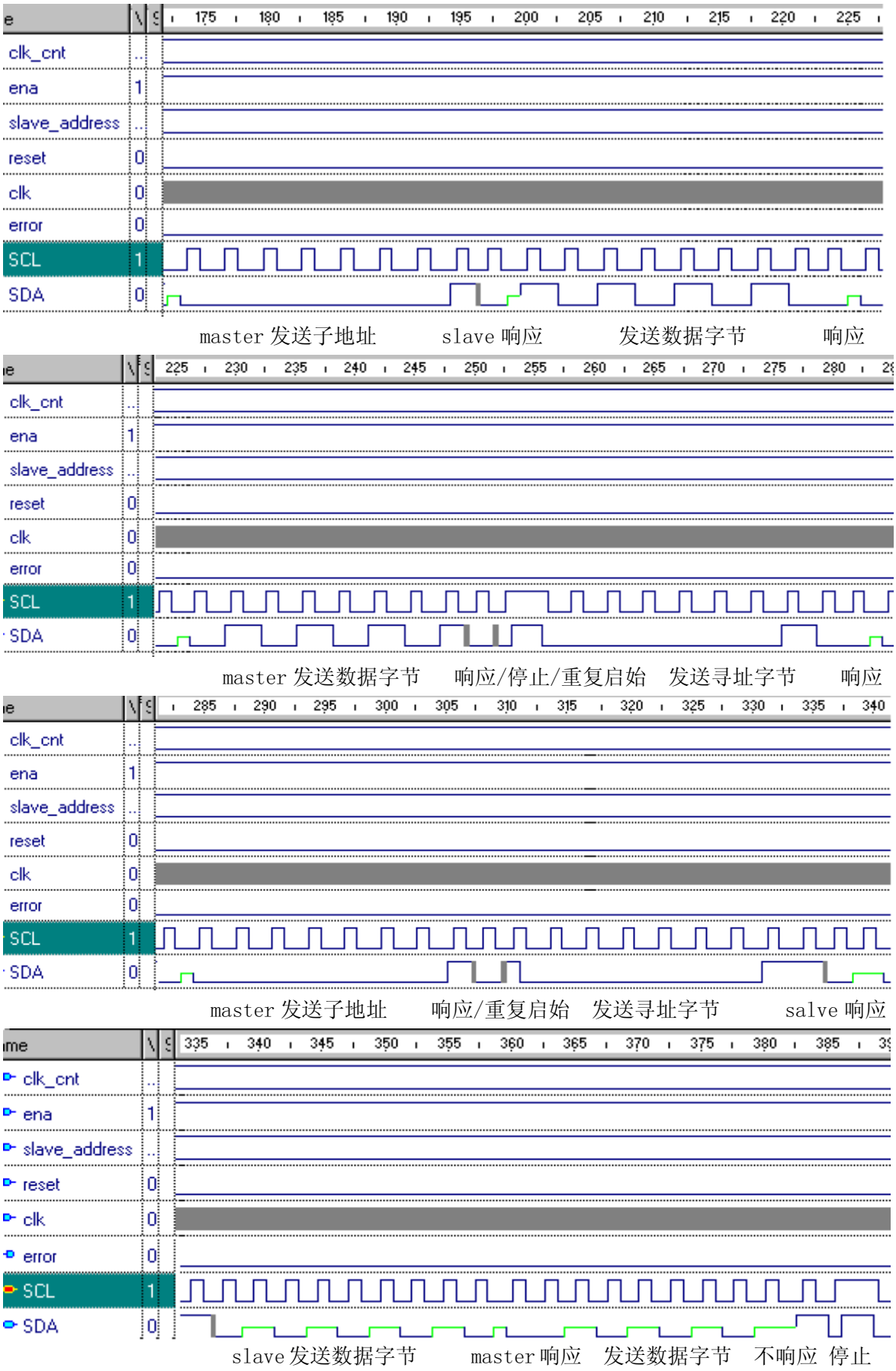


图 5-1 从模式的 I²C 总线接口电路的模拟结果

5.2 主从模式混合的 I^2C 总线接口电路的测试

测试过程使用两个该接口电路。在测试向量的激励下，一个被配置为 master 另一个则被配置为 slave， master 和 slave 之间再进行数据传输的测试。具体测试过程：首先初始化测试向量的控制信号，然后产生复位信号将接口电路复位,接下来开始 master 向 slave 发送数据的进程，进程结束后读取 master 状态字，最后开始 master 向 slave 读取数据的进程，如果进程结束，测试也就到此结束。

master 向 slave 发送数据的具体过程：首先测试向量内置的状态机向 master 和 slave 的各内部寄存器写数据,初始化发送进程,具体而言就是分配 master 和 slaveI^2C 总线地址、使能master和slave,将寻址字节写入master的数据寄存器等待发送。然后状态机向master的控制寄存器写入控制字，使 master 产生启始信号，开始进行寻址过程，状态机检测信号 MCF 信号判断寻址字节是否完成传送。传送完成，状态机向 master 的数据寄存器写入需要发送的数据，再检测信号 MCF 信号判断是否完成传送，如果数据传送完成，则再次向 master 的数据寄存器写入数据直到所有数据发送完毕，最后状态机向 master 的控制寄存器写入控制字，使 master 产生停止信号终止数据传送。

slave 向 master 发送数据过程：首先测试向量内置的状态机向 master 和 slave 的各内部寄存器写数据，初始化发送进程,再向 master 的控制寄存器写入控制字，使 master 产生启始信号，态机检测信号 MCF 信号判断寻址字节是否完成传送。传送完成，接下来 master 开始接收数据，所有的数据接收完成后， master 不响应 slave，以便产生停止信号。最后，状态机向 master 的控制寄存器写入控制字，使 master 产生停止信号终止数据传送。

VHDL 语言实现的测试向量：

测试中设置两个数组分别按顺序存放测试中的所使用的地址和数据，并用 cycle 作为指针，来寻址得到每个数据传送周期中测试所需要的地址和数据。

测试控制进程：

初始化控制进程

```
cycle <= 0;
go <= '0';
write <= '0';
```

一个数据传送周期

```
cycle <= cycle + 1;
wait until mcf = '1';      --只用于检测数据是否以被 I2C 控制器发送完毕
go <= '1';
wait until clk'event and clk = '1';
wait until clk'event and clk = '1';
go <= '0';
wait until done = '1';
```

测试内置的状态机：

该状态机是由时钟上升沿驱动进行状态转换的。

系统复位后状态机进入 IDLE 状态， AS， DS 输出为高电平，信号 done 为 1 。如果信号 go 变成 1,系统将信号 done 置为 ‘ 0 ’ ，并将地址发到数据总线上，如果是写周期，将

数据也发到数据总线上。如果信号 go 变成 1 ，状态机的下一个状态为 WAIT_DTACK 状态；如果信号 go 维持 0 状态，状态机也将继续保持 IDLE 状态。

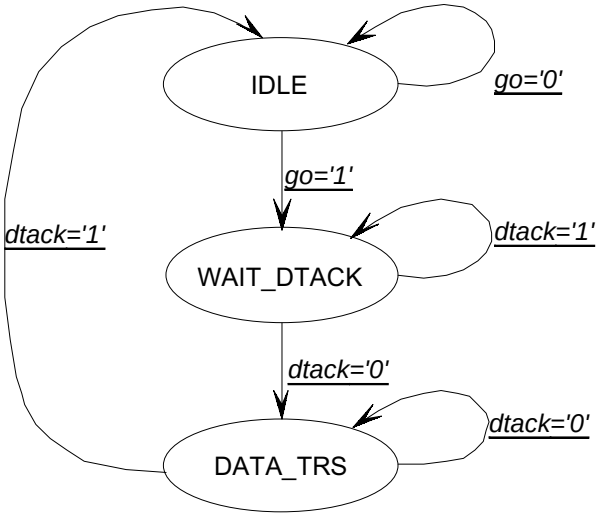


图 5-2 测试内置的状态机

在 WAIT_DTACK 状态， AS 、 DS 输出置为低电平,地址和数据的输出保持不变,信号 done 保持 0 状态。如果微处理器接口响应，即 dtack=0,状态机的下一个状态为 DATA_TRS ，否则保持 WAIT_DTACK 状态不变。

在 DATA_TRS 状态， AS ， DS 输出恢复高电平，信号 done 置为 0 ，地址输出保持不变，如果是写周期，数据输出也保持不变，如果是读周期，将数据线上由微处理器接口发送的数据锁存。如果 dtack 变成 1 ，状态机就返回 IDLE 状态；否则继续在 DATA_TRS 等待 dtack 变成 1 。

六、 VLSI 实现

整个设计是用 VHDL 语言进行行为级描述的，首先需要通过 Active-HDL 的功能仿真器的功能仿真。为了减少竞争冒险、提高电路运行的可靠性，所有的电路设计都采用同一时钟控制下的同步时序电路。例如，状态转化并没有依靠 SCL 时钟的下降沿触发，而利用高速时钟的上升沿采样 SCL ，检测到 SCL 时钟的下降沿后，在时钟的上升沿触发状态转化。这样就避免了由于总线负载过重而导致 SCL 下降时间太长所带来的一系列问题。功能仿真正确以后，再用 Synposys 的 Design Compiler 综合得到门级电路。这里我们所用的技术库是无锡上华的器件库（ csmc06core.db,csmc06core.sdb). 为了方便处理我们的逻辑设计与物理设计环境间的数据交换，我们加上 Design Compiler 的辅助工具 Floorplan Manager. 综合得到的门级电路网表经过适当修改，加上约束条件及 FTGS 或 VITAL 的库，再利用 Active-HDL 的时序仿真器进行时序仿真。时序仿真需加入 Design Compiler 生成的标准延时格式文件。时序仿真正确以后，再用 Candance 的 Silicon Ensemble 工具进行对 Design Compiler 生成的 Verilog 网表进行布局布线。在布局不布线之前需为 Verilog 网表加入最顶层的 PAD 说明，以及各层 pin 的说明。翻译成版图以后再次利用 Active-HDL 的时序仿真器进行版图后仿真。仿真时需要加入由 Silicon Ensemble 生成的标准延时格式文件。时序满足要求以后就可以去流片了。

七、结论

可配置的 FIR 滤波器具有极大的灵活性，但如果采用传统的总线接口电路，可配置的功能的实现会占用大量的宝贵的芯片管脚，而从模式的 I^2C 总线接口电路可以很好的解决这个矛盾。该从模式的 I^2C 总线接口电路通过行为级仿真、综合后门级时序仿真，其功能符合 I^2C 总线协议 ，可以很好的完成对可配置的 FIR 数字滤波器的参数配置和参数读回验证的任务。其小巧的结构也不会占用过多的芯片资源。尽管相对传统的总线接口，从模式的 I^2C 总线接口的速度慢一些，但系统配置不象系统运行时强调速度，所以采用从模式的 I^2C 总线接口，牺牲一些速度是值得的。

I^2C 总线的应用越来越广泛，很多 IC 设计生产商都推出了具有 I^2C 总线接口的 IC 器件，将原来具有并行接口的 IC 器件改造成具有 I^2C 总线接口的 IC 器件是很有意义的。但，如何改造呢？本文提出的主从模式混合的 I^2C 总线接口电路设计方案，为这种改造提供了便利，因为该接口具有一定的通用性，可以和器件原来的并行接口直接相连。该接口通过了行为级仿真、综合后门级时序仿真，其功能符合 I^2C 总线协议。