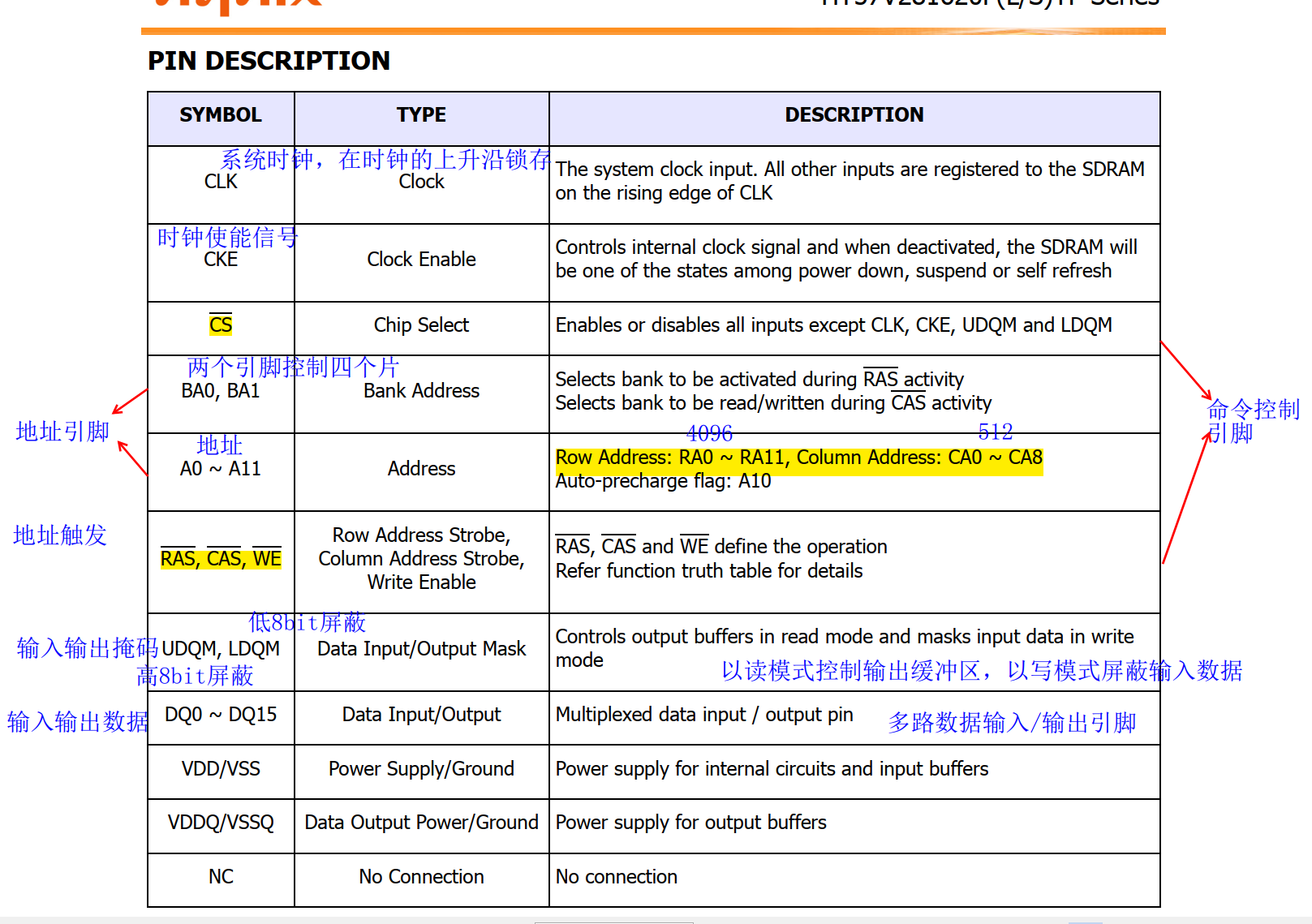
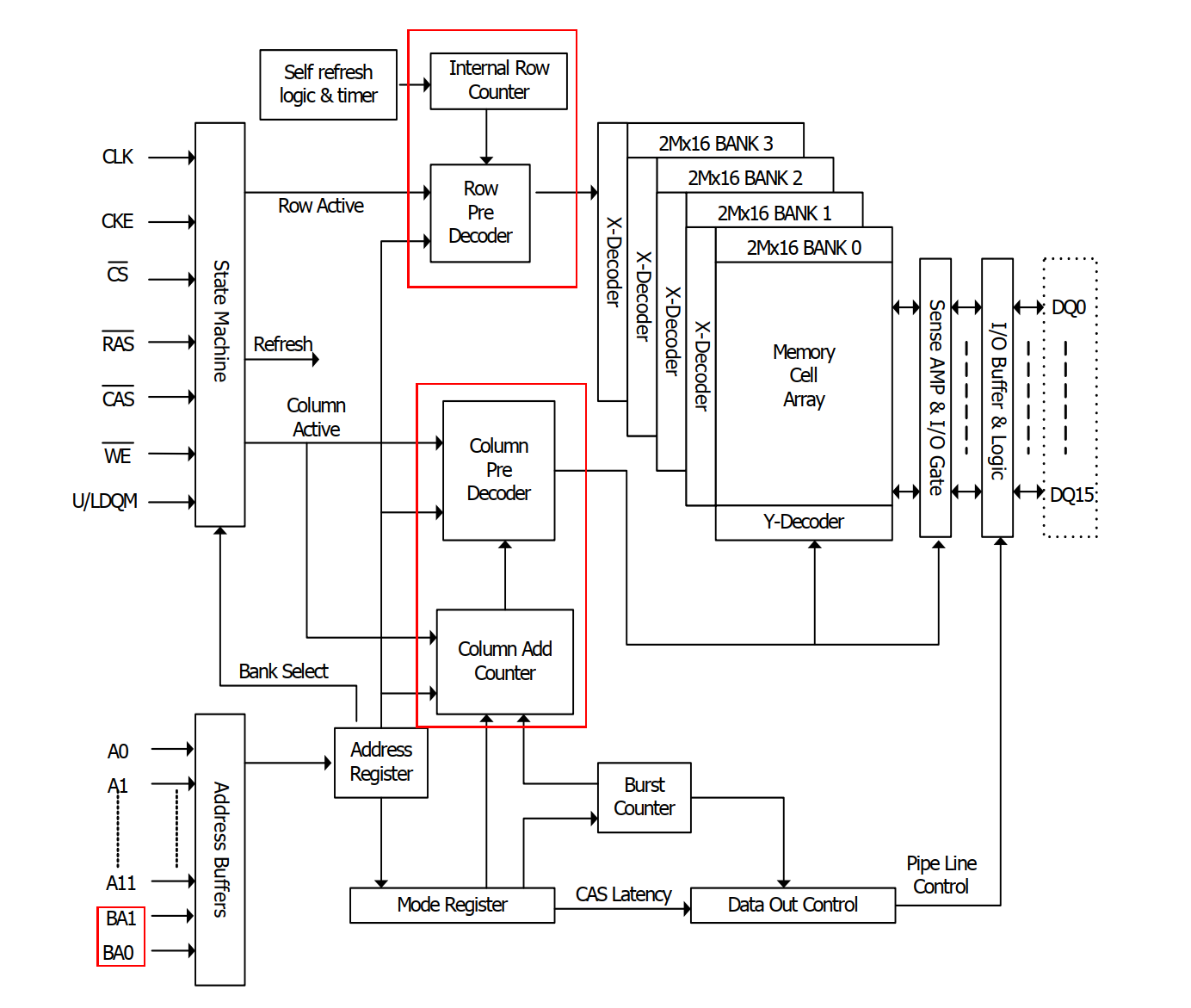
SDRAM controller设计文档

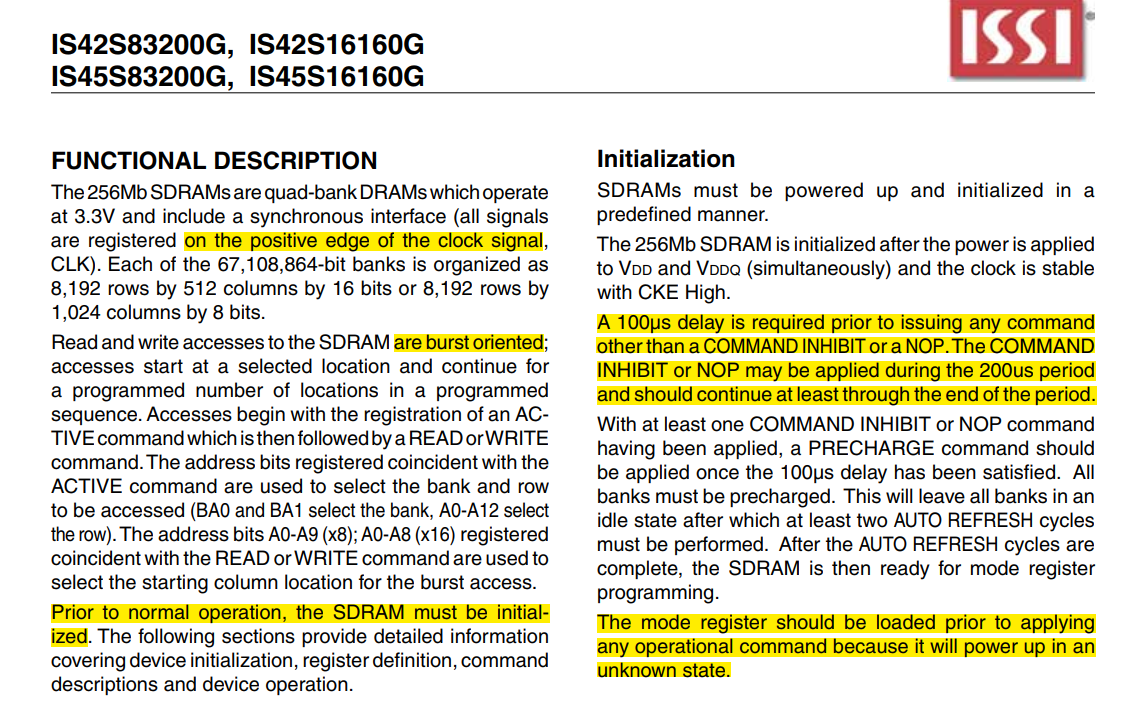
1. 总体概述





1. 初始化

1.简述：



初始化要在电源和时钟稳定

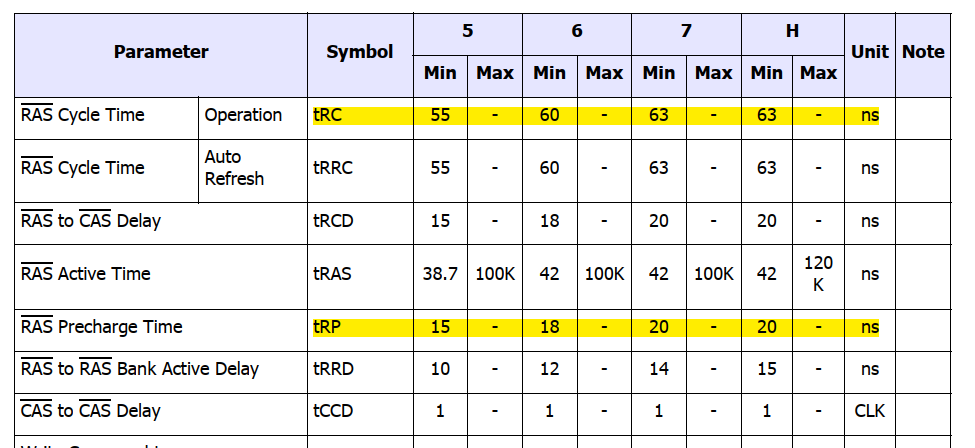
任何命令之前需要100µs无任何命令。COMMAND INHIBIT或NOP可以应用在200us期间，并至少持续到该期间结束。

至少有一个INHIBIT或NOP指令应用,100US满足后才能用percharge。所有bank都必须预percharge。所有bank会处于空闲状态，之后必须执行至少两个自动刷新周期。完成后，SDRAM就可以进行寄存器状态了

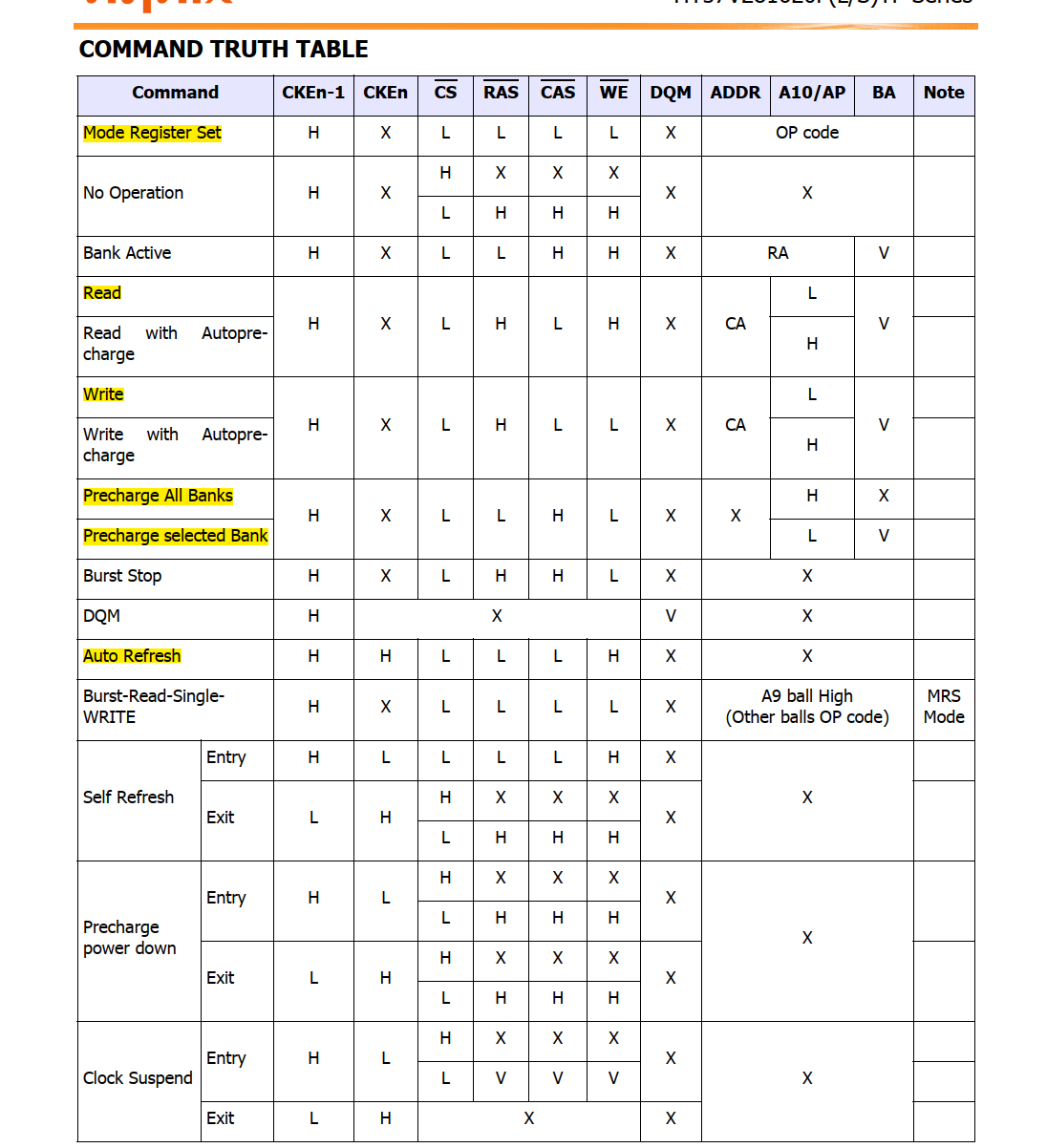
2.时序

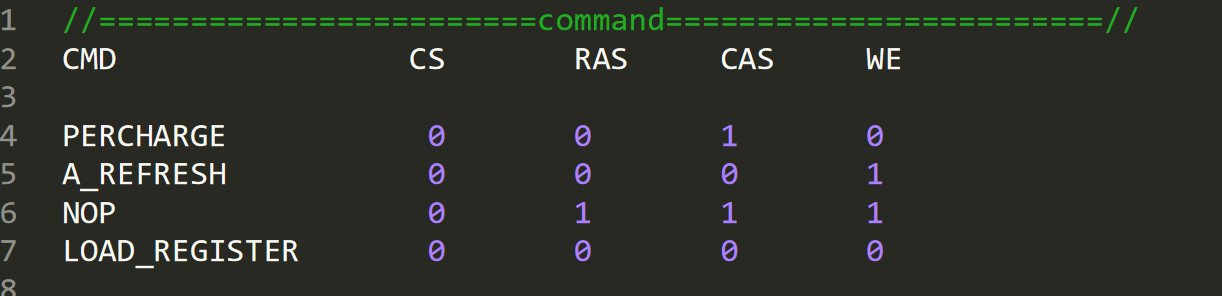


3.延时



4.command

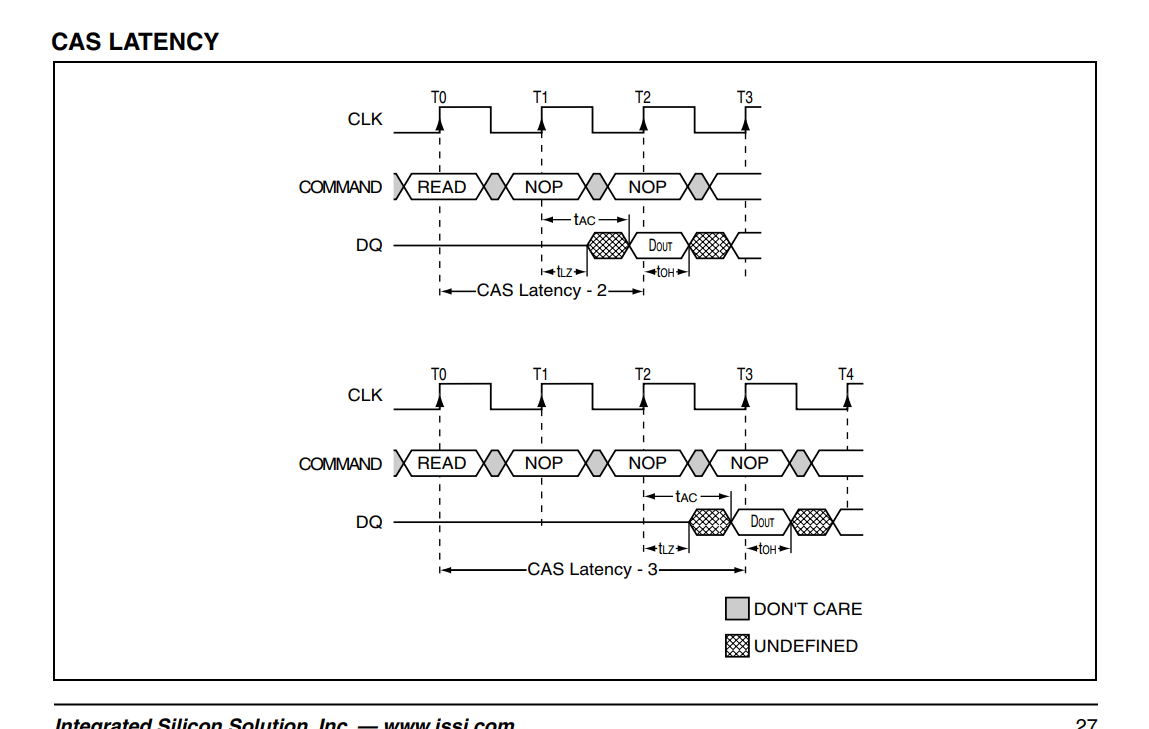


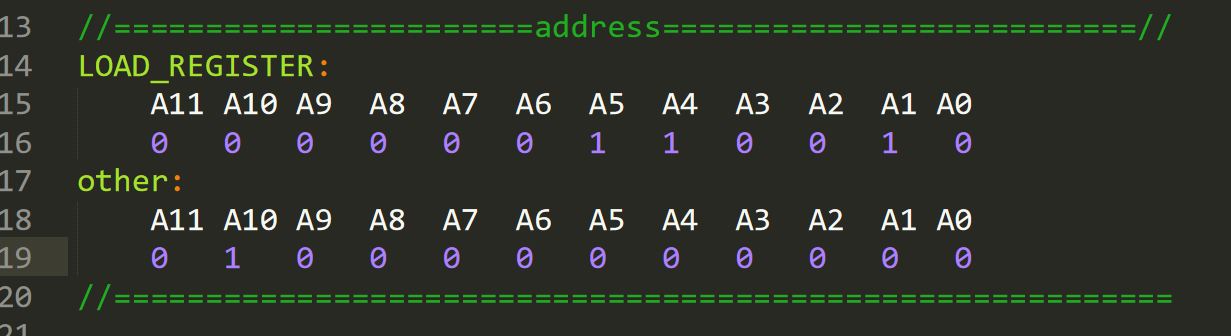


5.配置寄存器

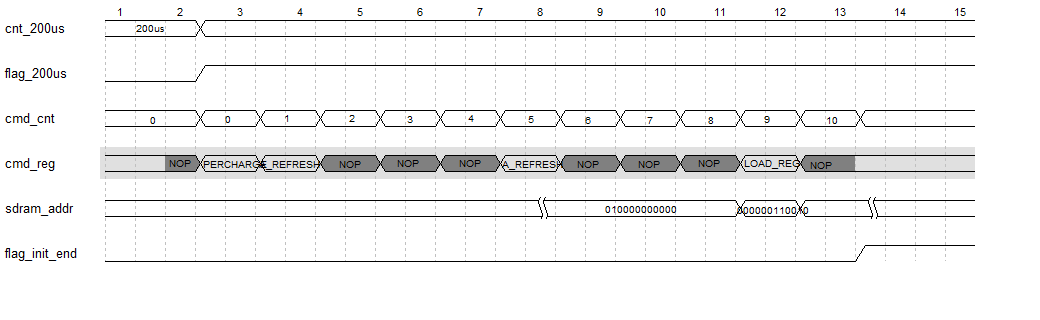


CAS信号：

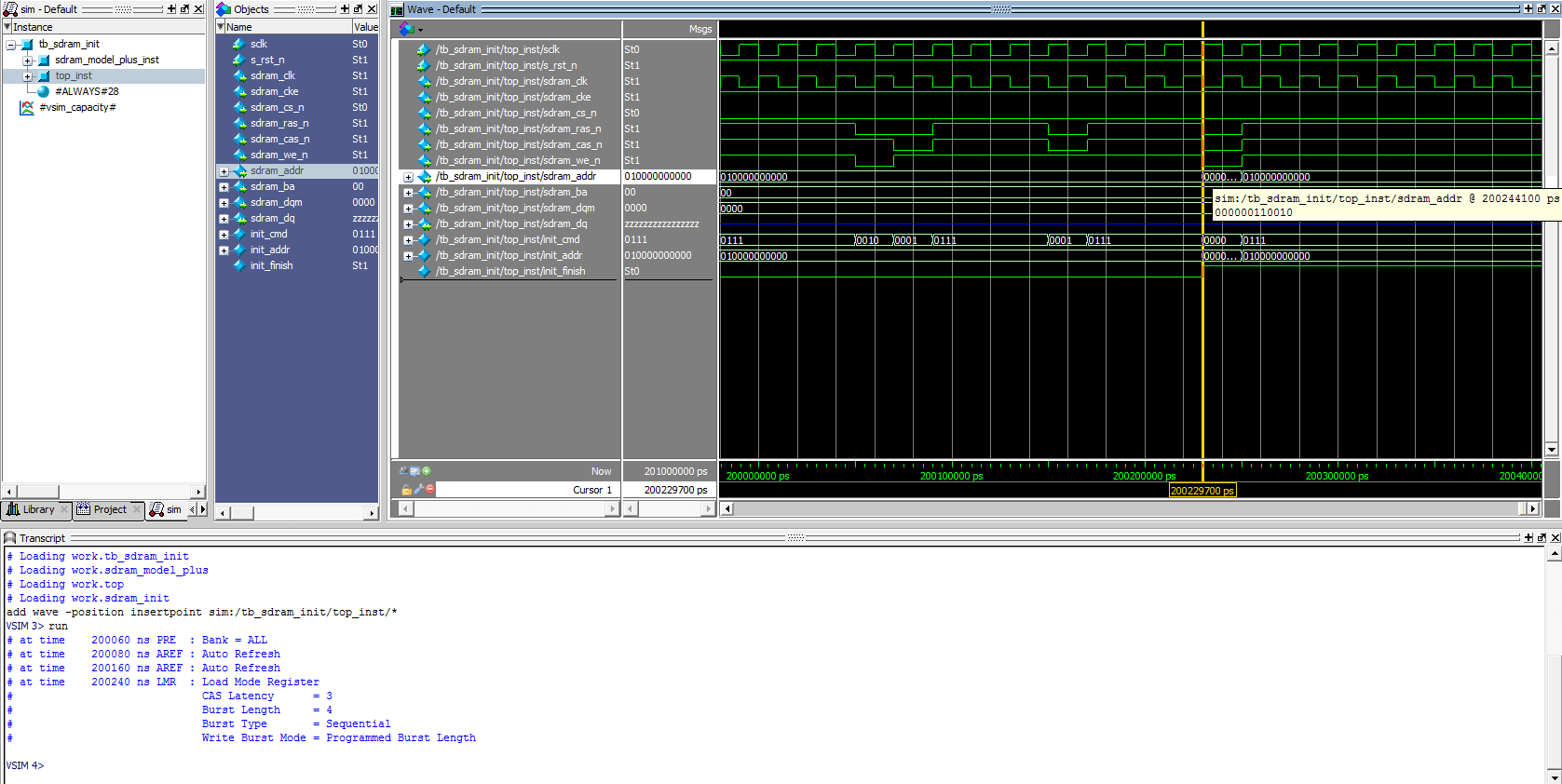


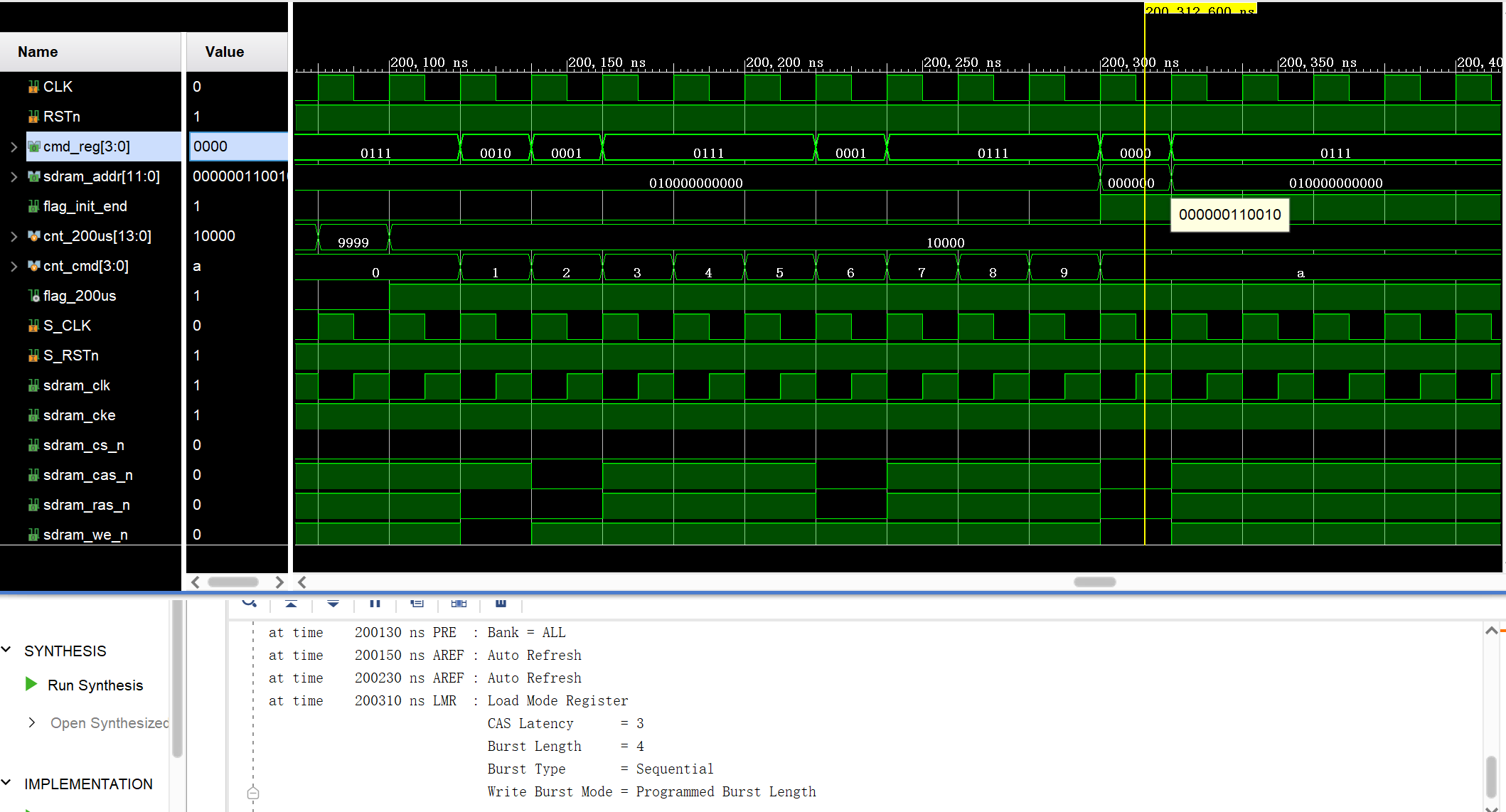


6.示意图

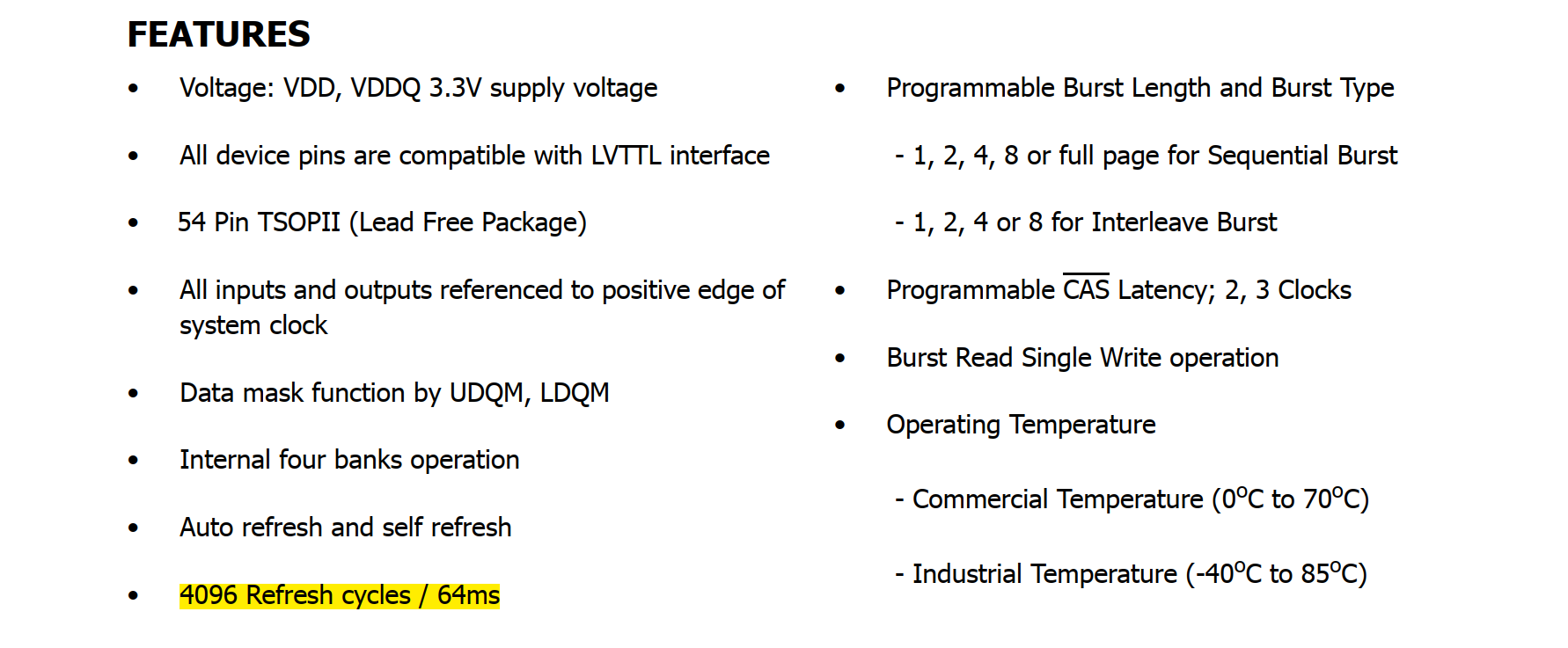


仿真结果：





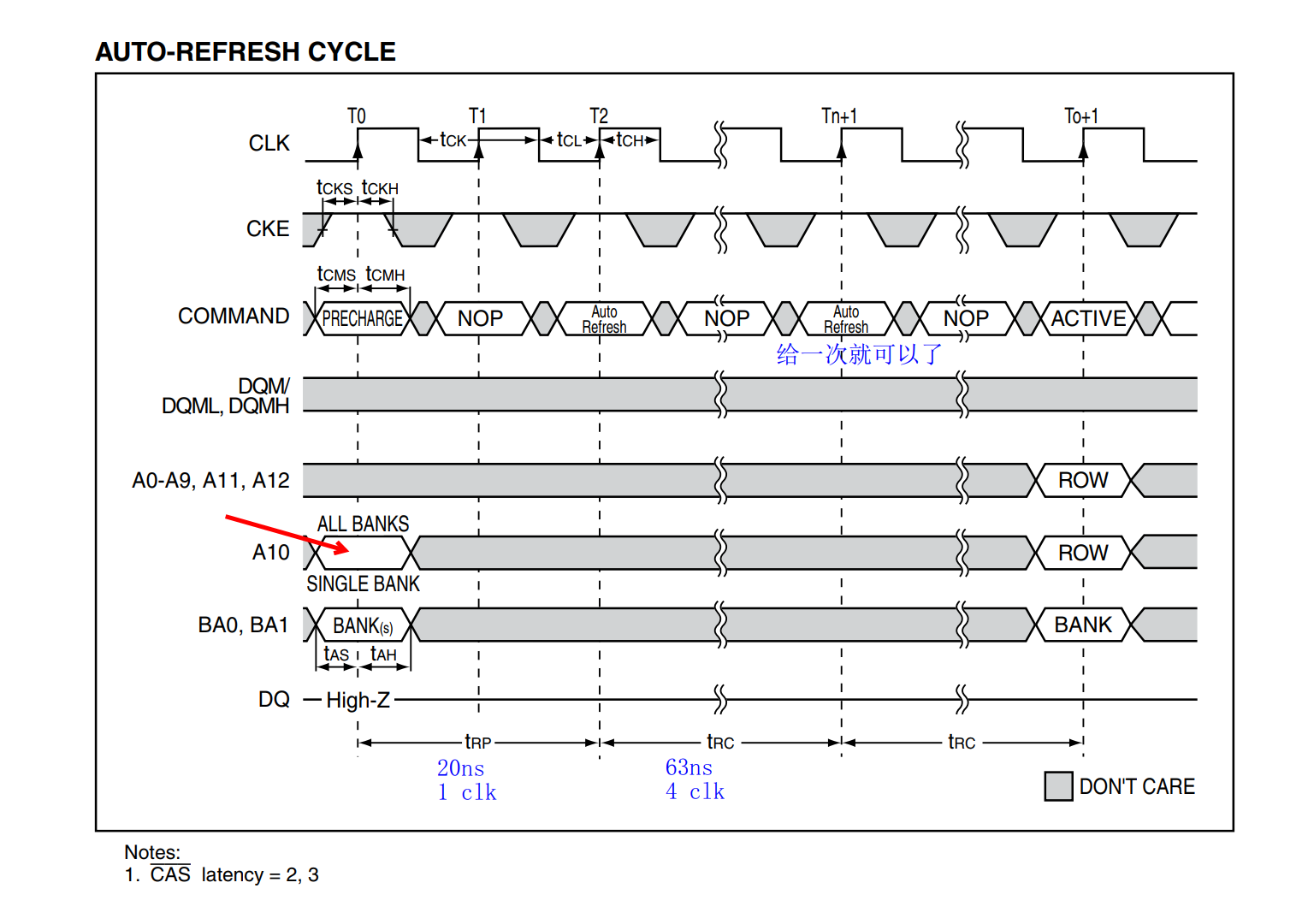
1. 刷新和仲裁
2. 简述



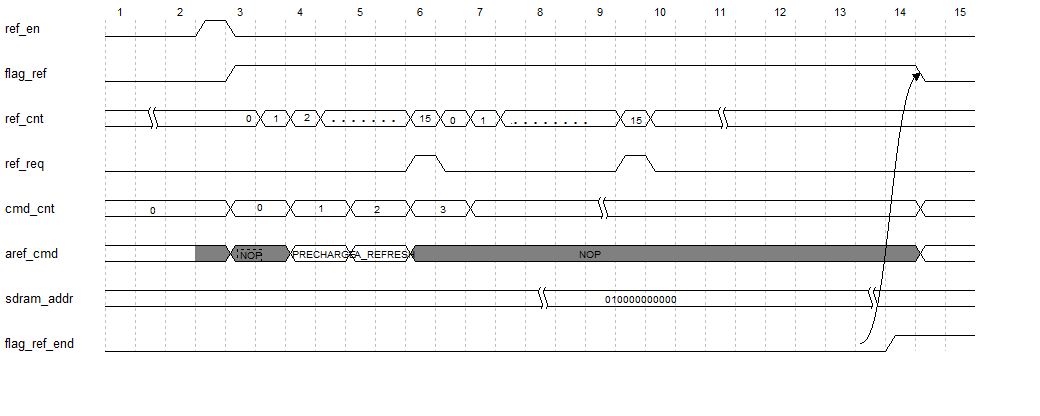
64ms是电容在未充电的状态下能保持电量的最长时间,也就是SDRAM在未刷新的状态下能保存数据的最长时间是 64ms。

4096,指的是在64ms的时间里边需要刷新4096次。因为 SDRAM 刷新是一行一行进行的，而控制行地址的总线有12bit，也就是一个Bank 有4096行。第一刷新第0行和第二刷新第0行的时间不能超过 64ms，但是需要在这个时间中刷新 4096 行，所以就需要刷新4096次了。所以，每两次刷新的时间间隔大约是64ms/4096=15.625us。可以取15us

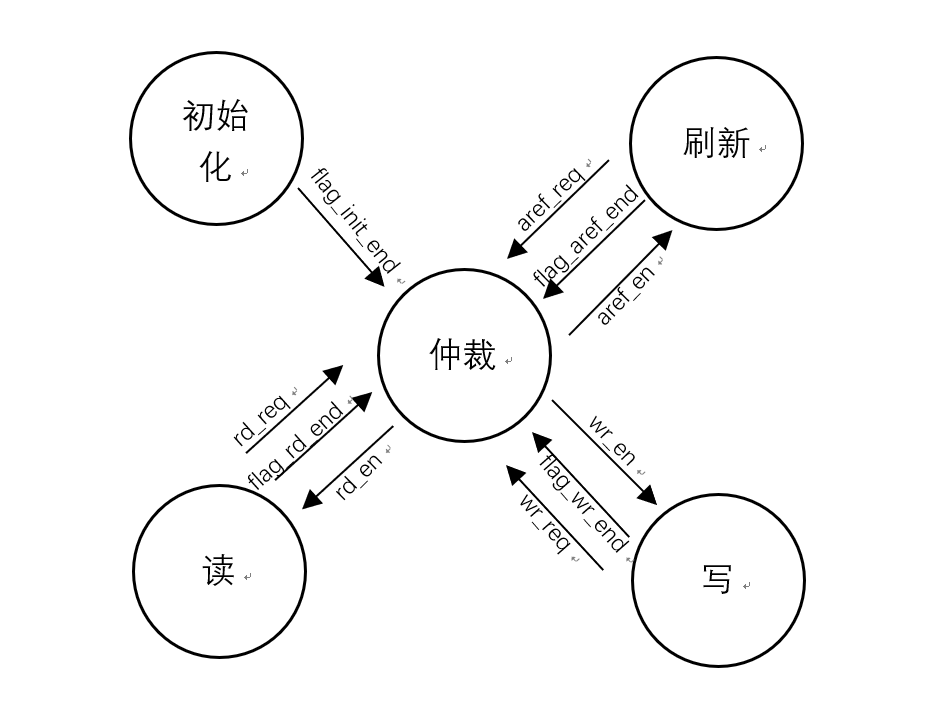
2.时序



3.示意图

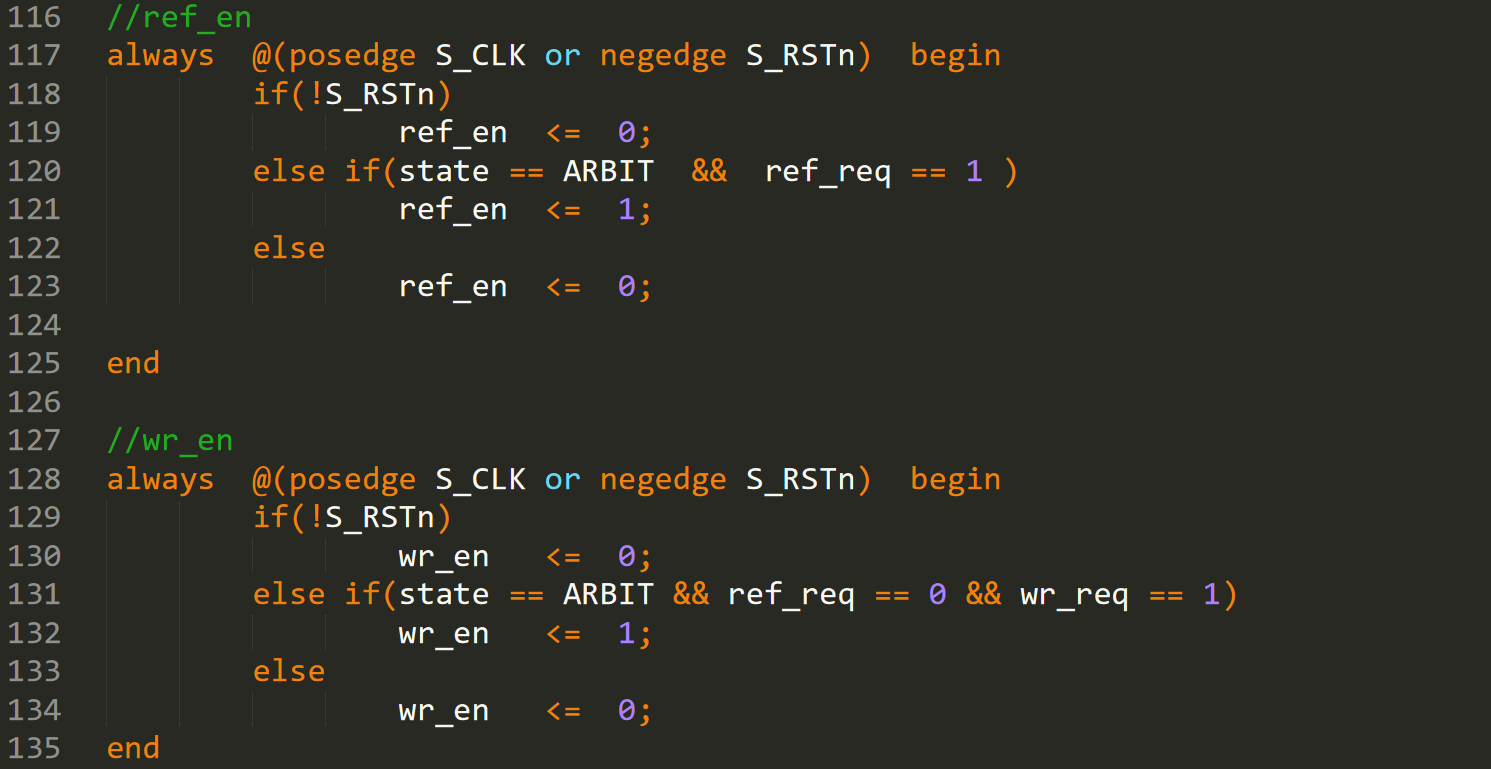


4.仲裁

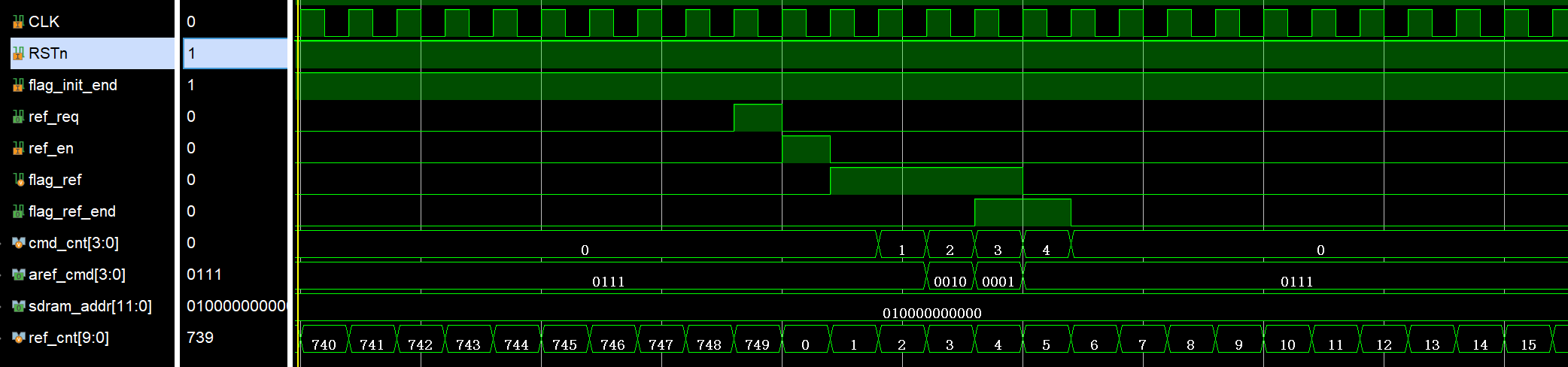
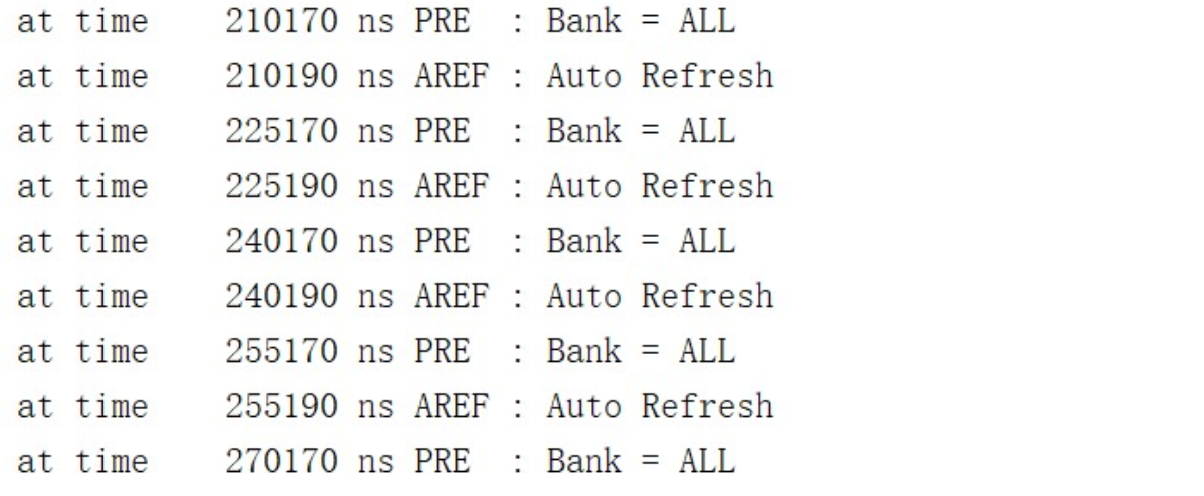
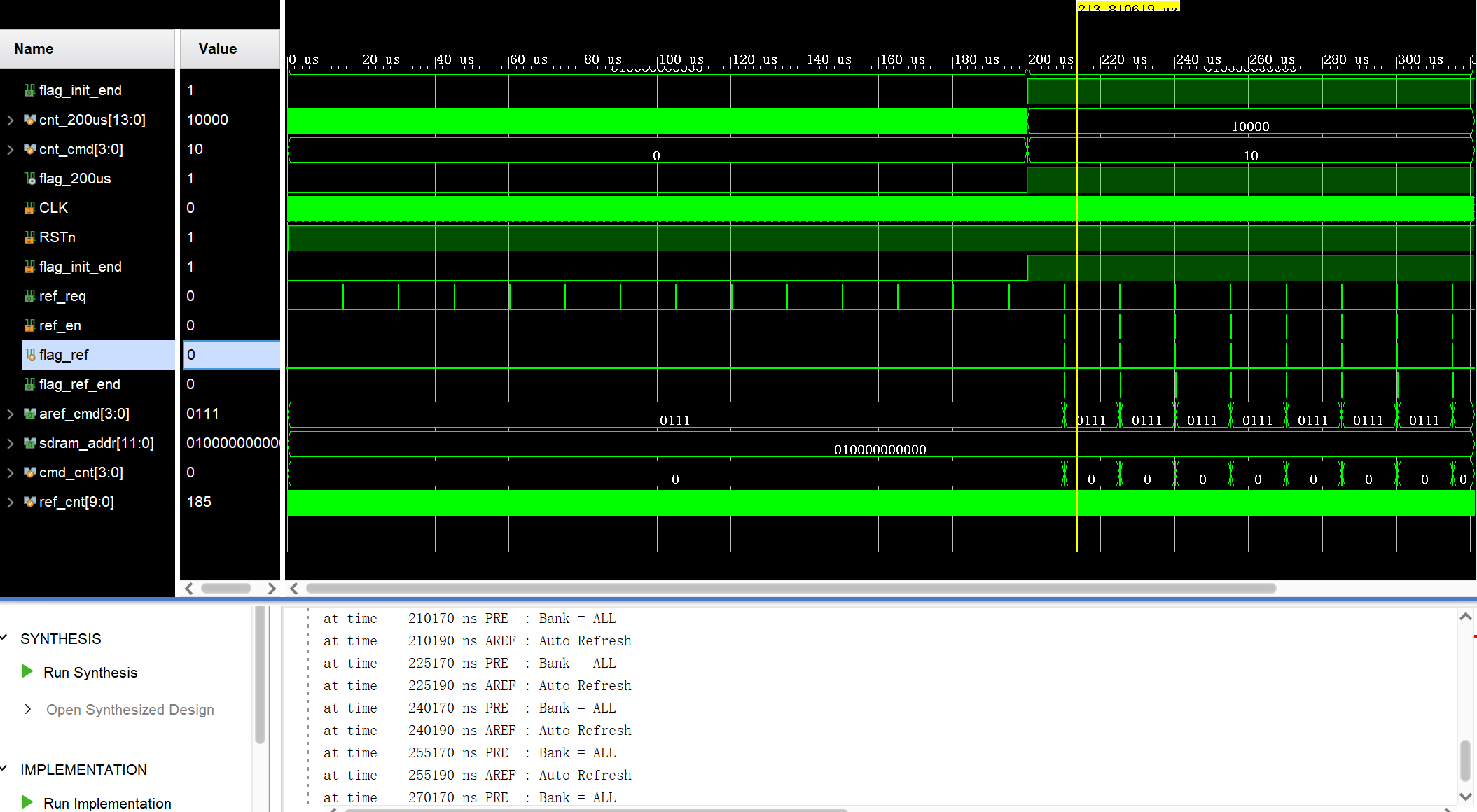


这就是最后top里的状态机。

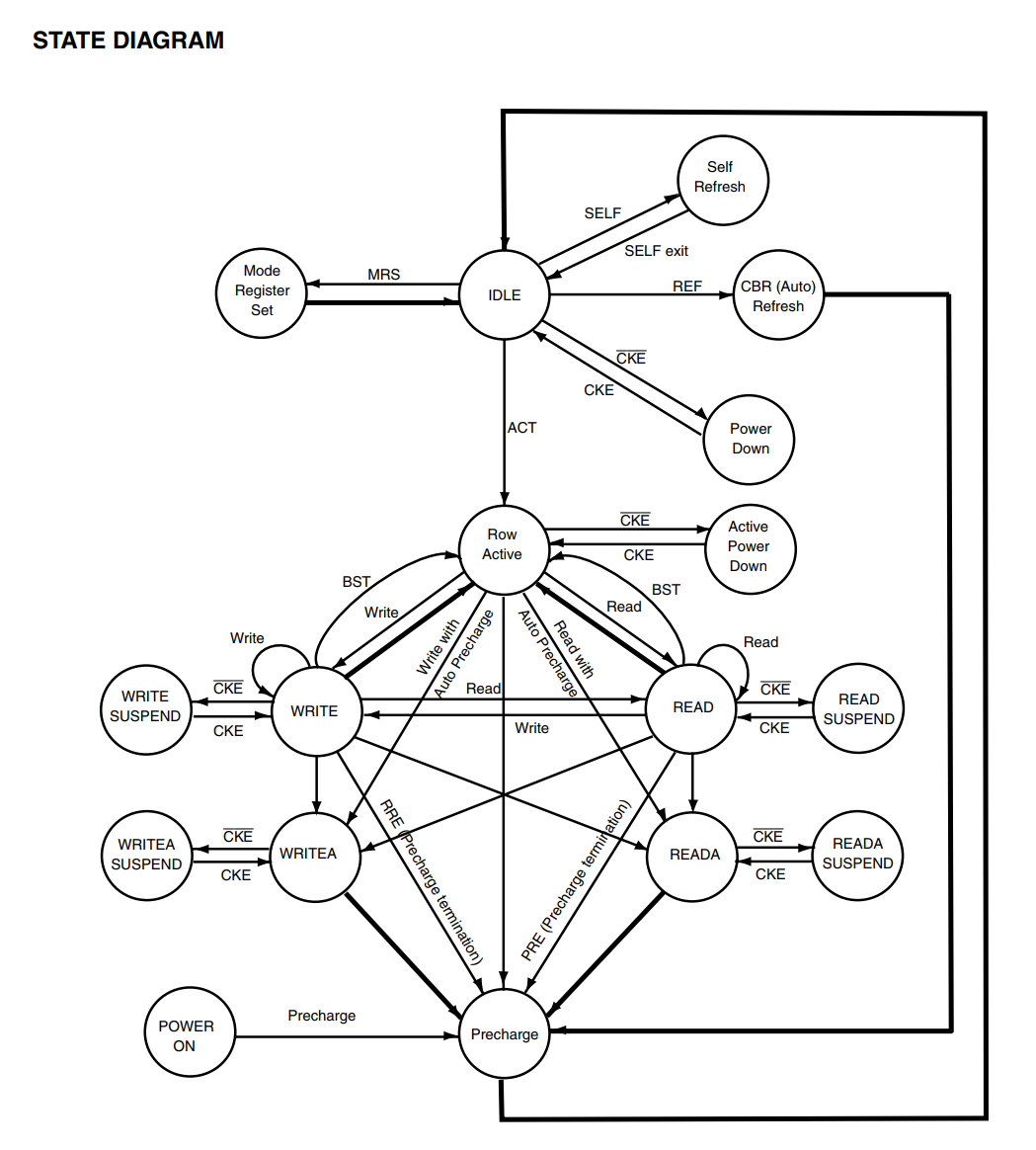
仲裁直接写在top中：



5.仿真结果



1. **读**/写
2. 简述



这是SDRAM器件本身工作的流程图了，开始于“Power On”状态，即 SADRAM 上电完成的状态。粗线表示自动跳转，细线表示需要执行相应的命令才会跳转到下一个状态。需要重点关注从“IDLE”状态跳到“WRITE”状态的路线，以及从“WRITE”状态跳到“IDLE”状态的路线。

**IDLE状态到WRITE状态：**

1)在IDLE状态需要先给ACT命令激活某一行，此时处于Row Active状态；

2)在Row Active状态之后，给 Write命令则会进入WRITE状态；

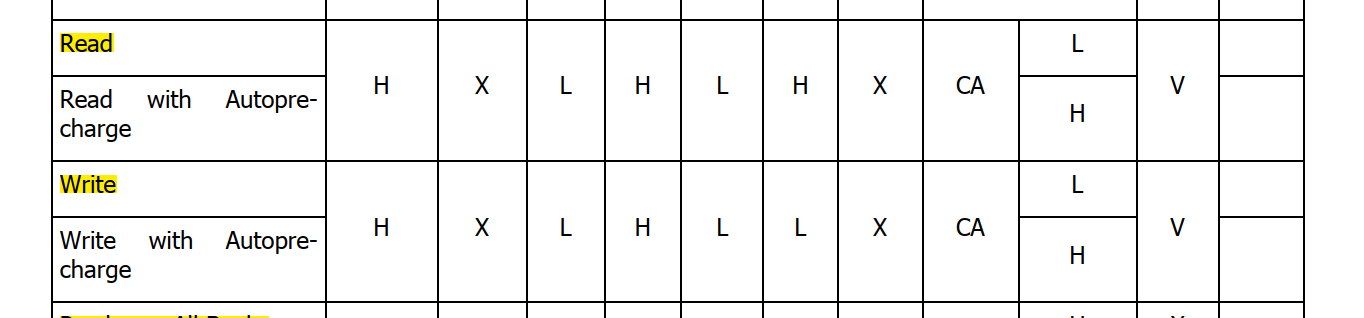
3)在WRITE状态后，再给一次Write命令，就可以继续写入数据。

**WRITE状态到IDLE状态（比如需要刷新，这就必须回到IDLE）：**

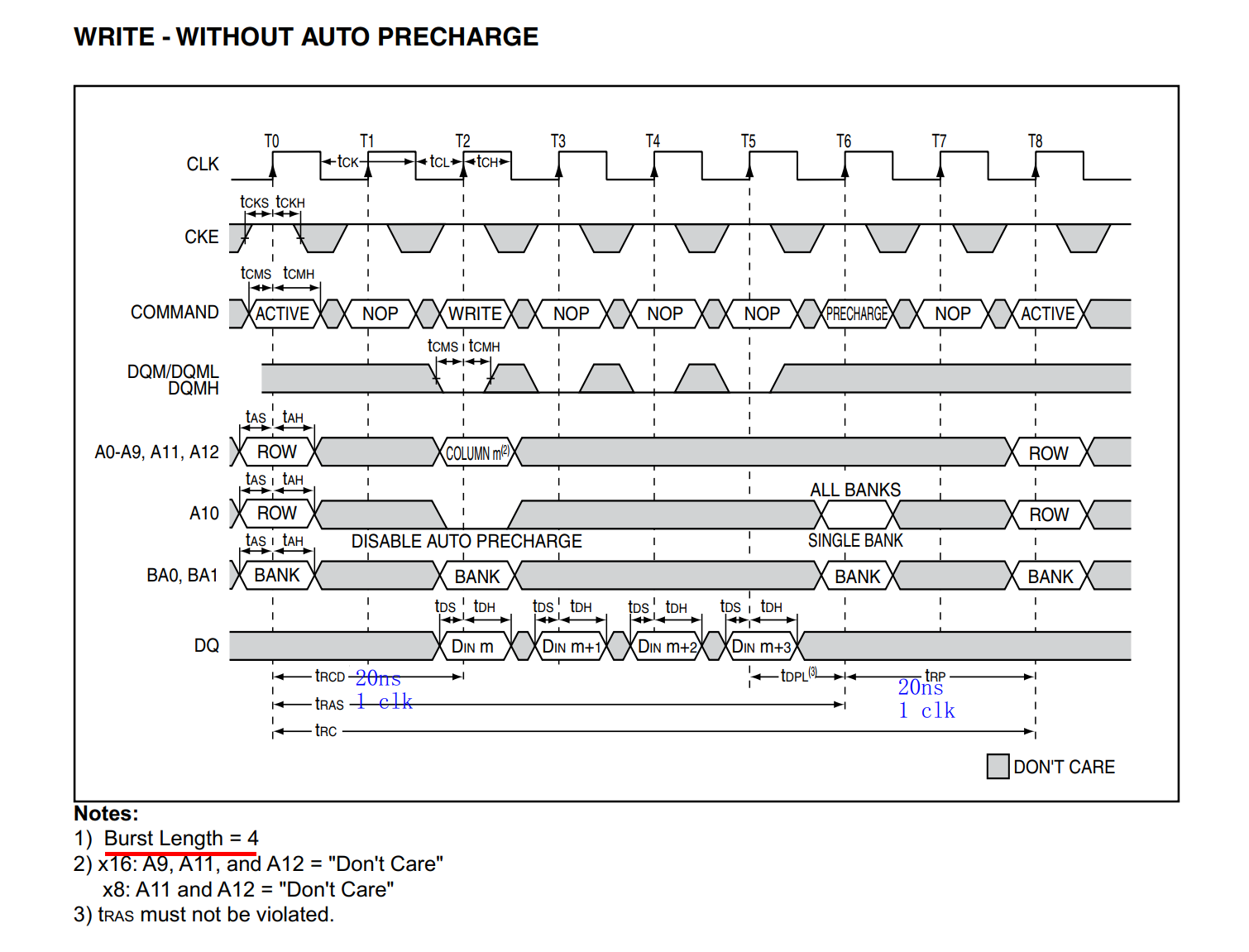
1)在WRITE状态给PRE命令，则SDRAM将跳出WRITE状态进入Precharge状态；

2) 在Precharge状态后，就会自动进入IDLE状态了。

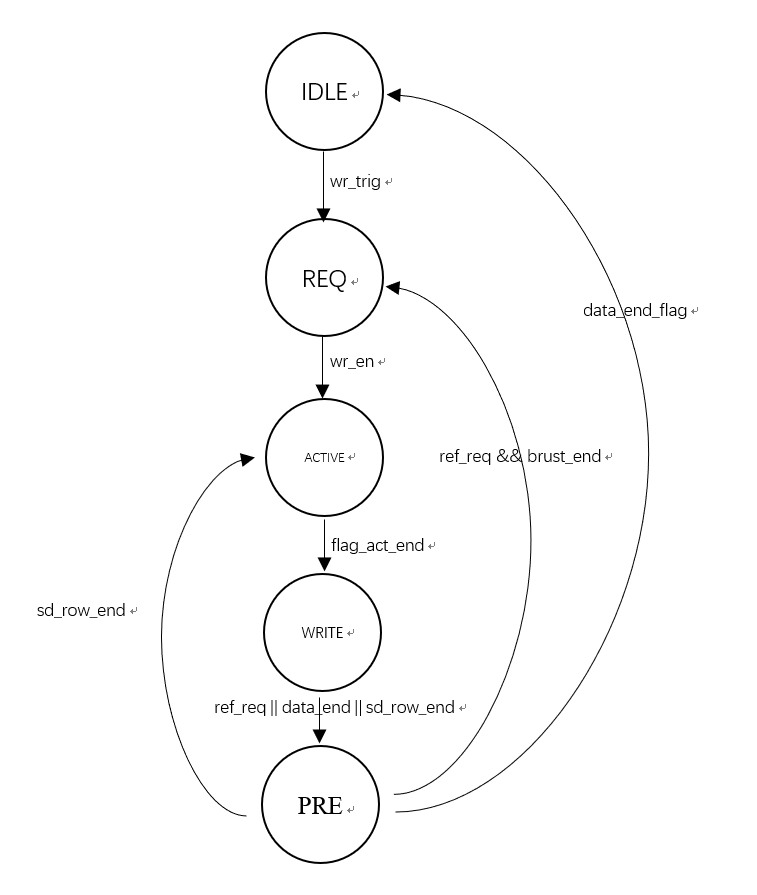
另外，WRITE状态下边还有一个WRITEA状态，他会自动的进入到Precharge状态。也就是说WRITEA比在WRITE状态的工作效率要低很多，所以使WRITE（without percharge）状态。



2.write without percharge时序



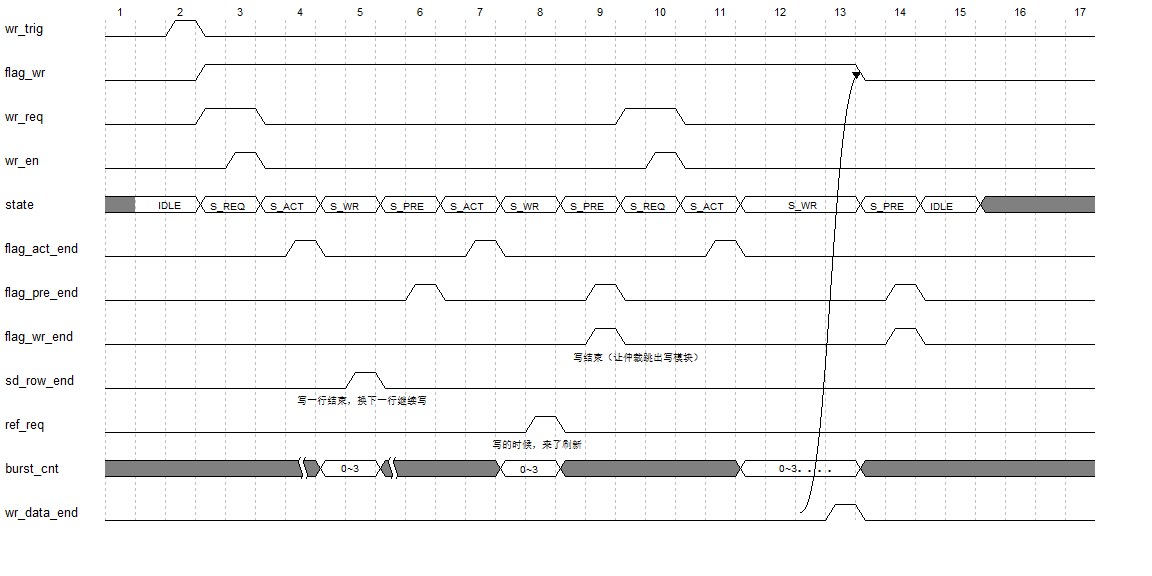
3.写模块内部状态机（写入两行数据）

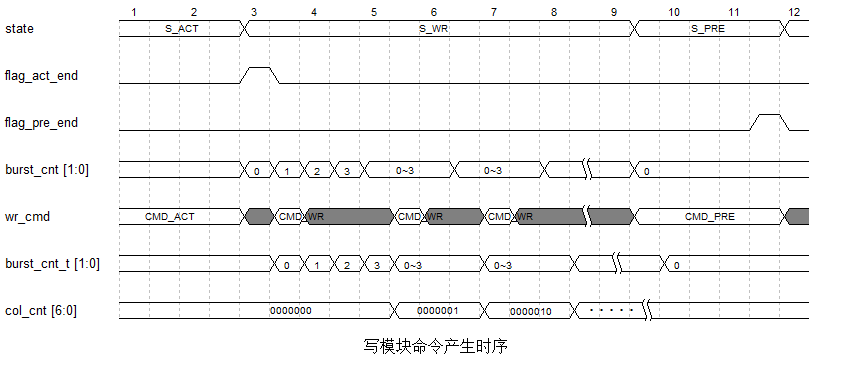


退出写状态：

1. 数据已经写完
2. SDRAM需要刷新
3. 数据未写完，需要激活下一行进行继续写（active时是已经定一行了）

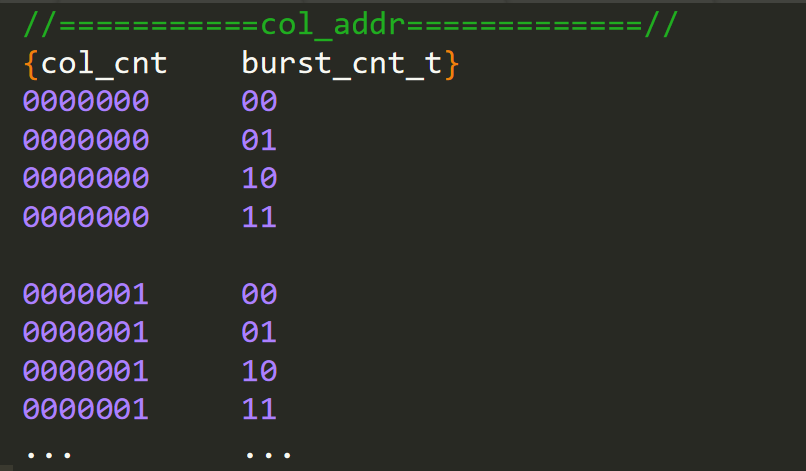
4.实现（写入两行数据的时序）



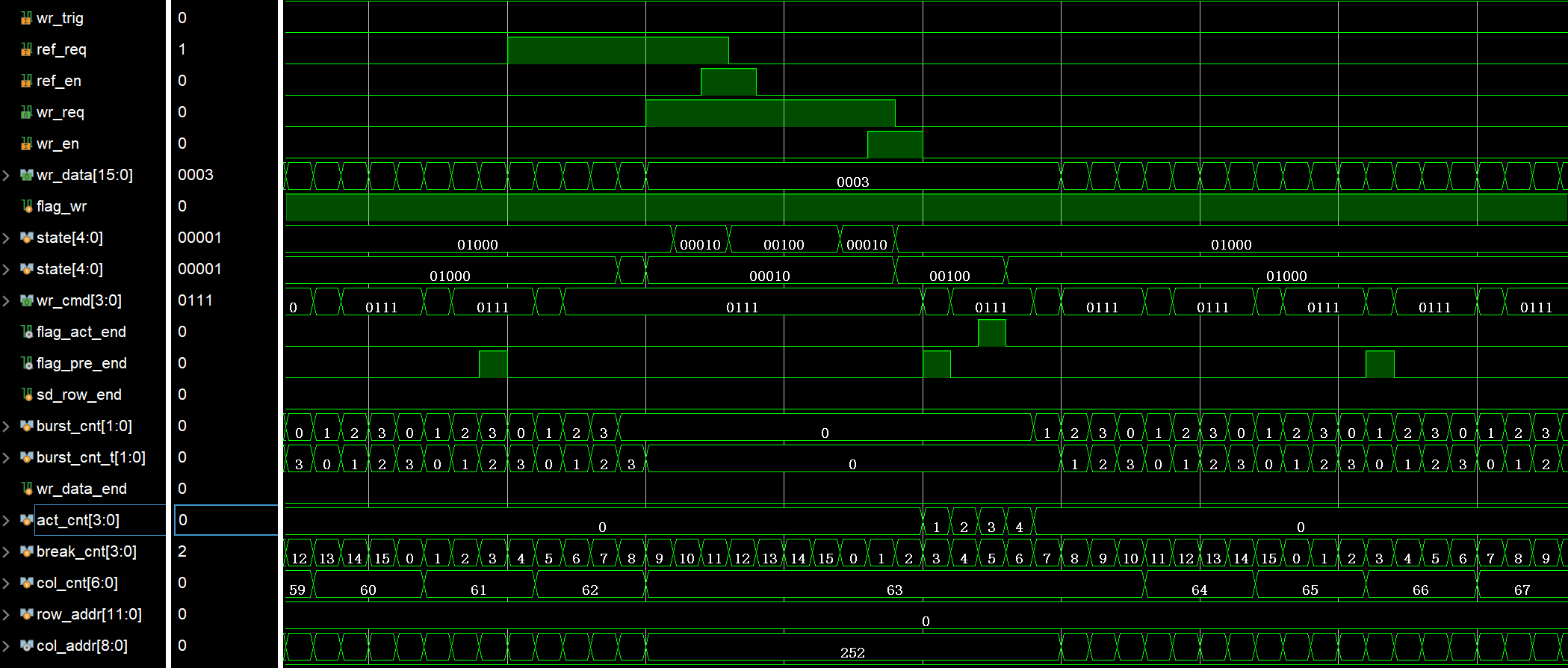
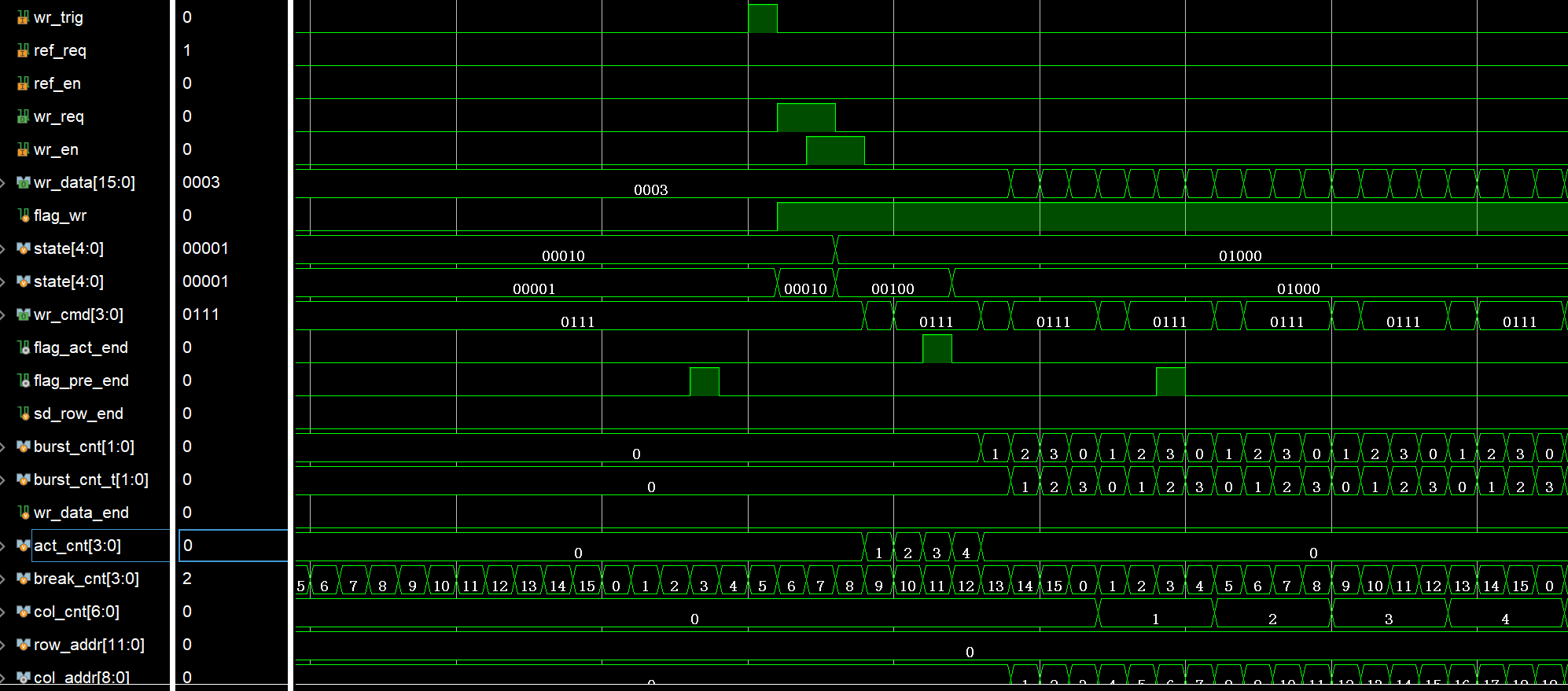
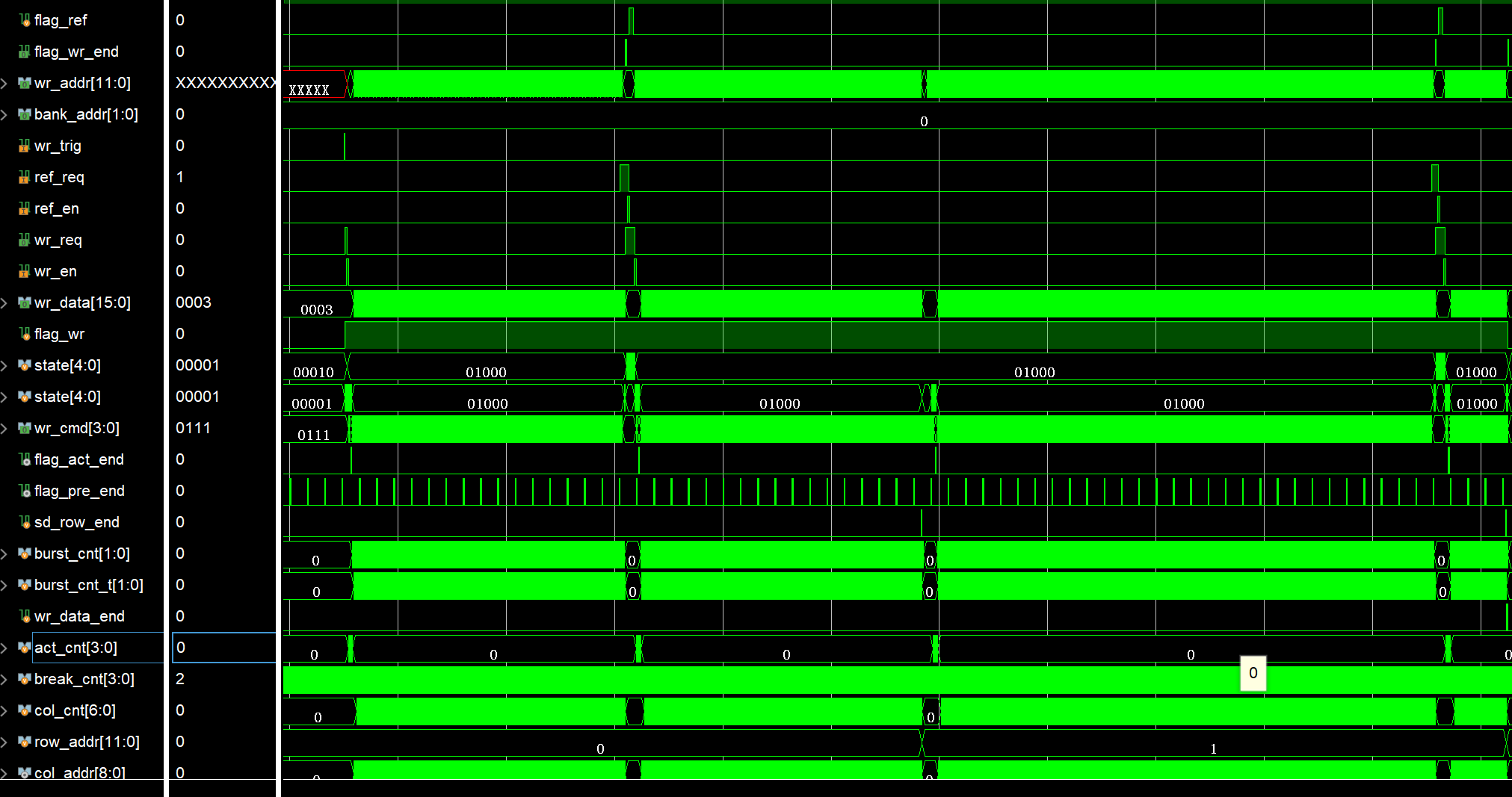
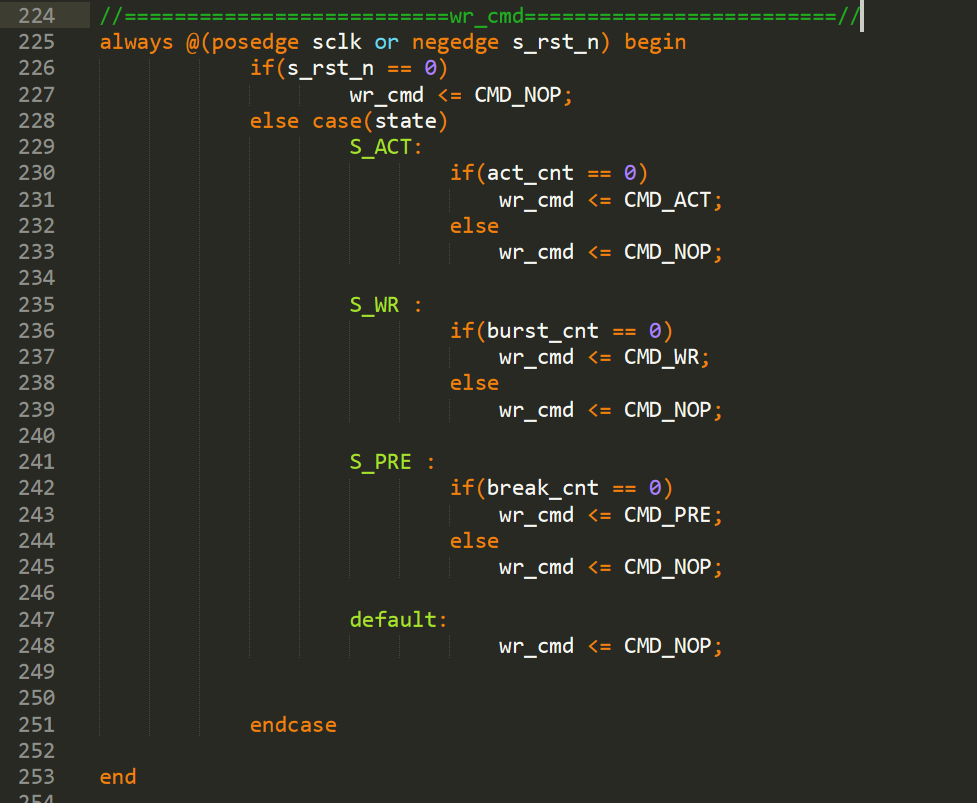


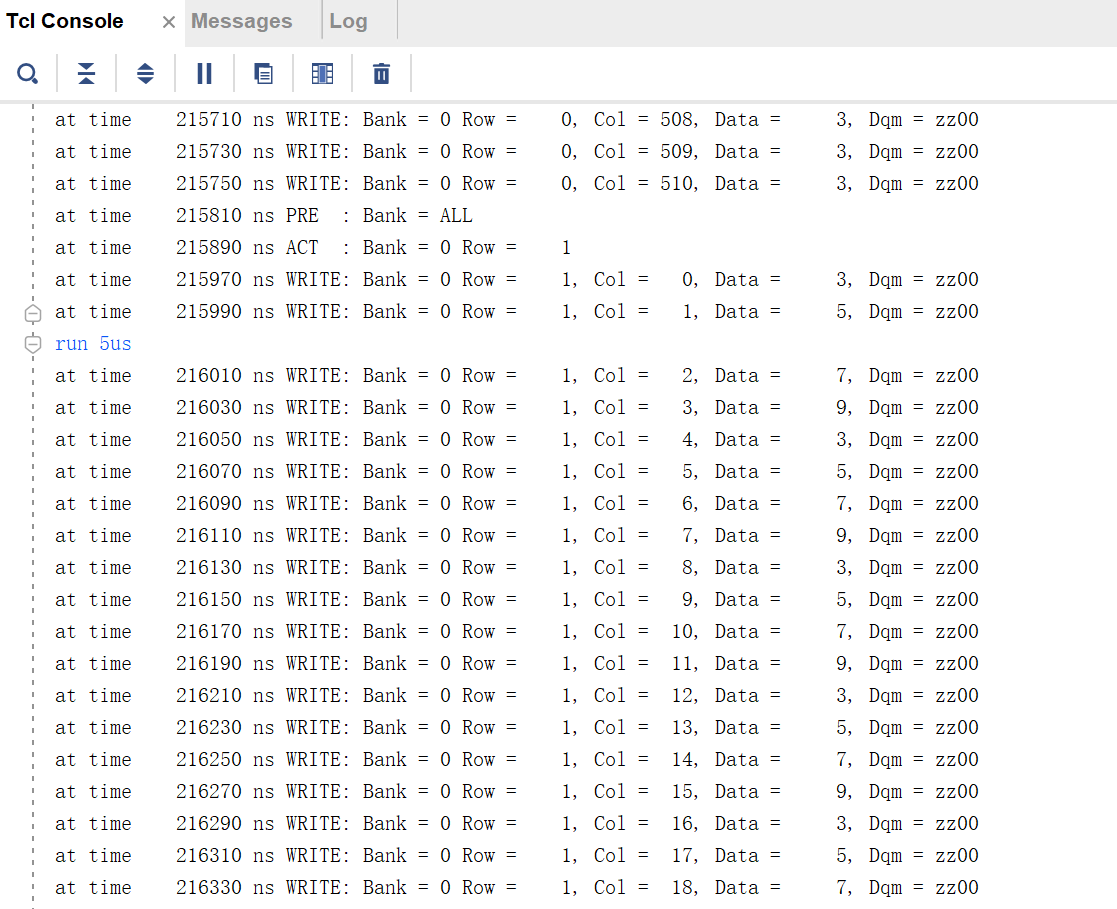
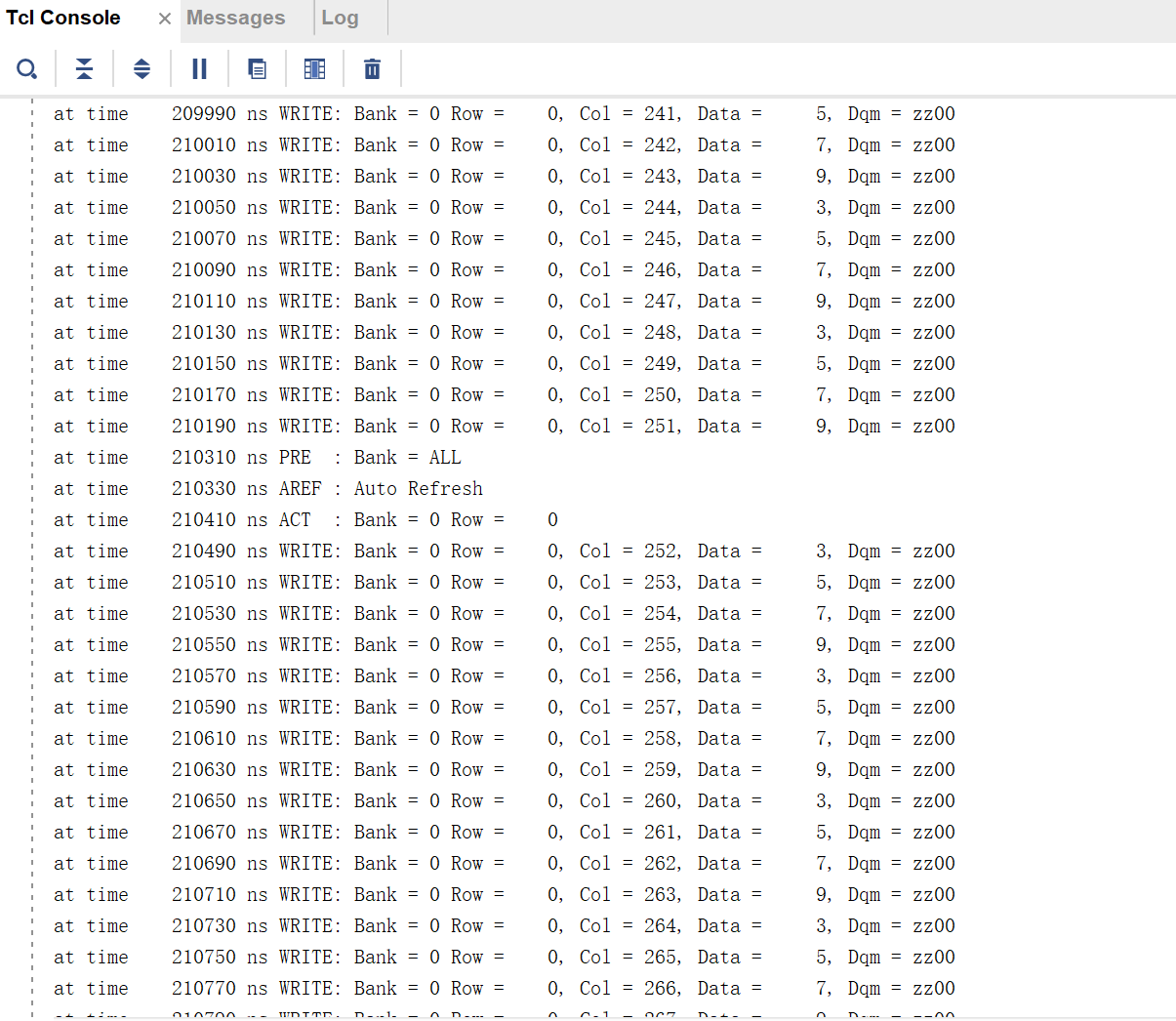
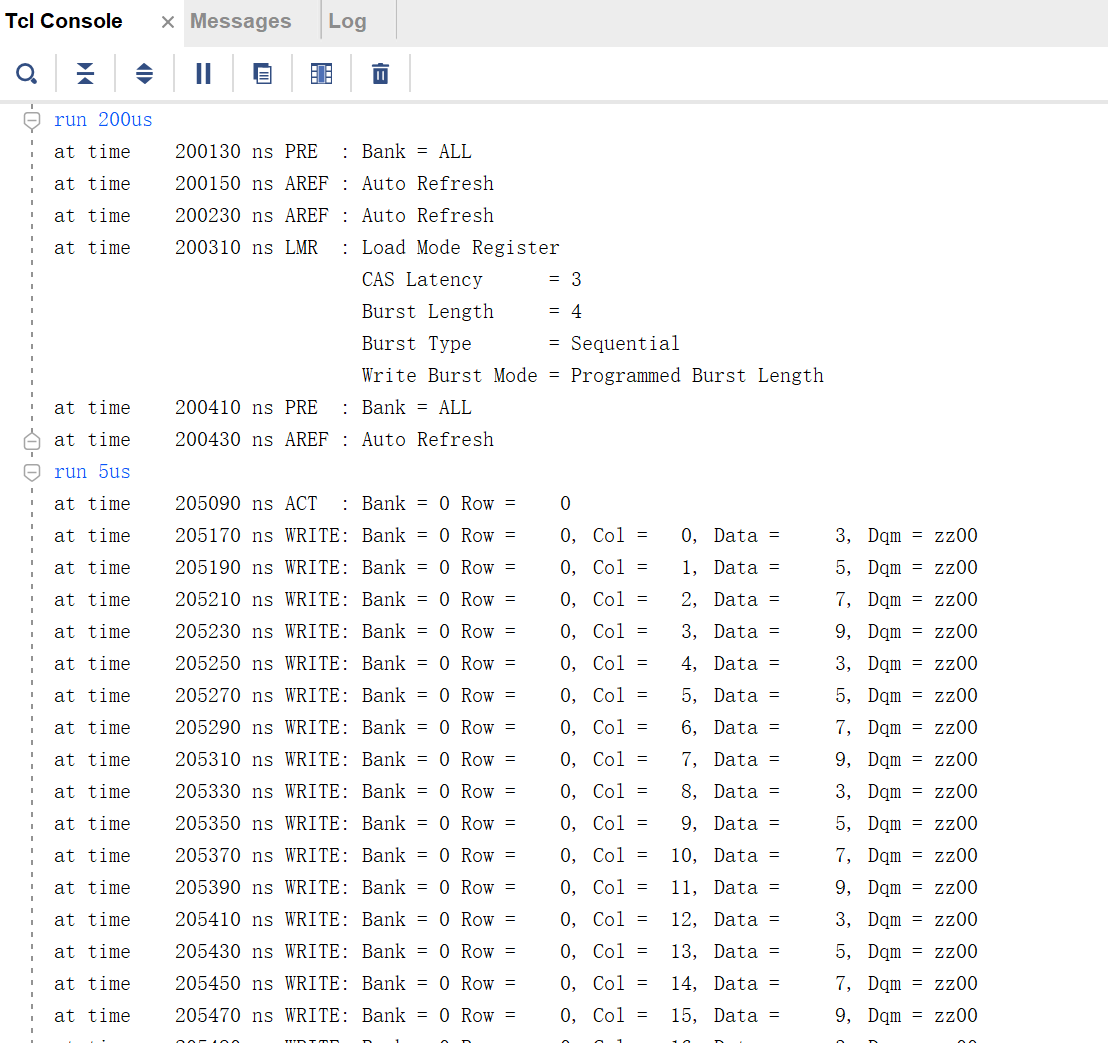
1. **地址：**

col\_addr = { col\_cnt, burst\_cnt\_t };



**（2） 命令：**





* 1. 读/**写**

同读模块

