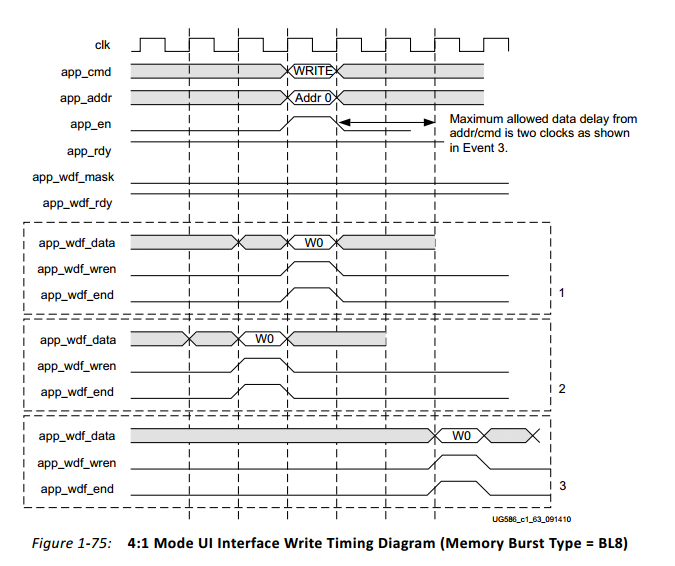
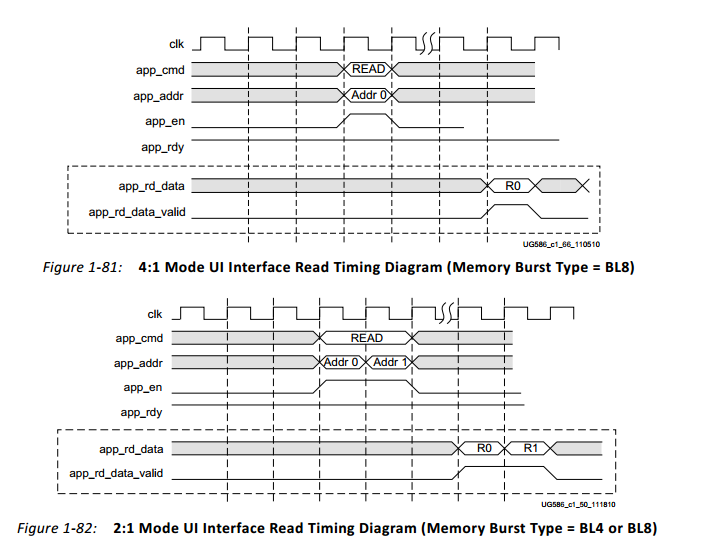
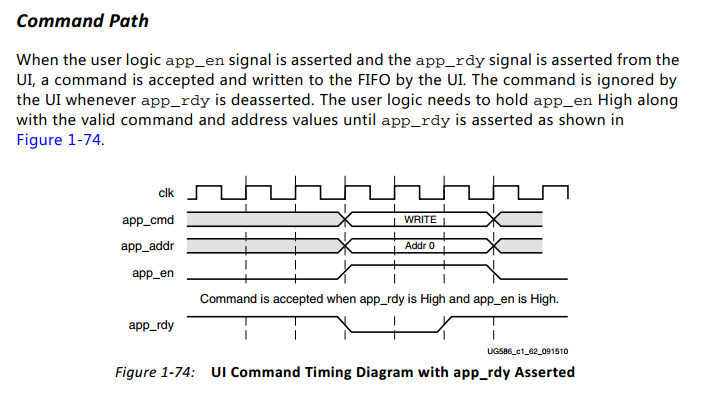
Mig 接口

|  |  |  |
| --- | --- | --- |
| 接口名称 | 方向（相对于mig来说） | 说明 |
| app\_addr[N:0] | 输入 | 当前响应给定的地址（15+10+3） |
| app\_cmd[2:0] | 输入 | 当前响应给出的读写命令，001：read  000：write |
| app\_en | 输入 | 该信号是app\_addr，app\_cmd，app\_sz，app\_hi\_pri信号的使能信号，当app\_en有效时，上述信号才能开始使用 |
| app\_rdy | 输出 | 该信号是准备好接收命令信号，当该信号为0时，且app\_en有效时，mig不会接收命令和地址，只有app\_rdy和app\_en同时有效时，mig才能接收命令和地址 |
| App\_hi\_pri | 输入 | 高优先级指示信号 |
| App\_rd\_data[n:0] | 输出 | 当mig是读命令时候，mig输出的ddr数据 |
| App\_rd\_dara\_end | 输出 | 当该信号为高时，表示当前输出的rd\_data是最后一个数据，只有app\_rd\_data\_vailed有效时候，该信号才能有效 |
| App\_rd\_data\_vailed | 输出 | 当该信号为高时，表示正在读数据 |
| App\_sz | 输入 | 该信号为保留位，使用时候给0就行 |
| App\_wdf\_data[N:0] | 输入 | 要写入的数据 |
| App\_wdf\_end | 输入 | 该信号为高时，表示当前数据为最后一个要写入的数据 |
| App\_wdf\_mask[N:0] | 输入 | 给app\_wdf\_data提供一个标记（直接给0就行） |
| App\_wdf\_rdy | 输出 | 该信号表示writefifo准备好接收要写入的数据了，当app\_wdf\_rdy = 1 && app\_wdf\_wren = 1的时候，开始写入数据 |
| App\_wdf\_wren | 输入 | 该信号是app\_wdf\_data使能信号，高电平表示可以写数据 |
| App\_correct\_en\_i | 输入 | 该信号一直给1就行 |
| App\_sr\_req | 输入 | 一直给0 就行 |
| App\_sr\_active | 输出 | 输出保留位（自己的逻辑中不用处理这个信号） |
| App\_ref\_req | 输入 | 当该信号高有效时候，表示刷新命令 |
| App\_ref\_ack | 输出 | 当该信号有效时候，表示mig给ddr发送了一个刷新的命令 |
| App\_zq\_req | 输入 | 高有效：表示zq计算命令是有问题的 |
| App\_zq\_ack | 输出 | 高有效：表示mig给ddr发送一个zq计算的命令 |
| Ui\_clk | 输出 | Ui时钟是dram时钟的一半或者四分之一 |
| Init\_calib\_complete | 输出 | Ddr初始化成功标志，高有效 |
| Ui\_clk\_sync\_rst | 输出 | 输出一个复位信号，高有效 |



读写状态机设计

驱动mig接口另外需要一个时钟ip，产生一个时钟，时钟ip提供一个migIp的200M时钟和一个100M的系统时钟

设计的状态机一共有六种状态，第一个状态是等待ddr矫正完成，当ddr矫正完成，状态机计入第二个状态，写状态，在写状态过程中，等MIG的ready信号接收命令和数据。一旦app\_rdy和app\_wdf\_rdy信号拉高，mig ip已经准备好接收命令和要写入的数据，当数据写完之后，直接跳转到写完成状态，在写完成状态中，app\_en和app\_wdf\_en拉低，当命令和数据线拉低后，跳转到读状态，再找个状态中，状态机等待mig的ready信号，当app\_rdy拉高，直接跳转到读完成状态，在读完成状态中，状态机拉低命令使能，同时等待app\_rd\_data\_valid拉高，ddr中数据就被读到了。同时跳转到检查状态，检查写入数据与读出数据是否相同，相同就点亮led，不同就不点亮，同时状态跳转到IDLE，等待下一次读写

