关于 ALTERA 提供的 FIFO 核使用原理

ALTERA 提供了 LPM_FIFO 参数宏模块，可以在代码中例化使用。

FIFO 有两种工作模式：（1）SCFIFO，（2）DCFIFO

其中 SCFIFO 指读写用一个时钟进行同步，可以支持同时读写的功能。

其中 DCFIFO 指读写使用不同的时钟进行同步，这在设计多时钟系统中相当有用，可用于不同时钟同步信号之间的同步调整。

首先看看 DCFIFO 模式下的几个比较重要的信号：

[A]在写端，主要有以下几个信号：

(1) data[n-1:0]：写入数据信号总线；

(2) wrreq：写入请求信号，高有效

(2) wrclk：写入同步时钟；

(3) wrfull, wrempty：用于指示写端 FIFO 为空或者满的状态；

(4) wrusedw[log2(SIZE_FIFO)-1:0] ：写入的数据个数，按写入个数递增；

上述信号都与写入时钟 srclk 同步；

[B]在读端，主要有以下几个信号：

(1) q[n-1:0]：读取数据信号总线；

(2) rdreq：读取请求/确认信号，高有效

(2) rdclk：读取同步时钟；

(3) rdfull, rdempty：用于指示读端 FIFO 为空或者满的状态；

(4) rdusedw[log2(SIZE_FIFO)-1:0] ：读取

FIFO 主要有两种工作模式：
(1) Legacy mode(Legacy synchronous FIFO mode)

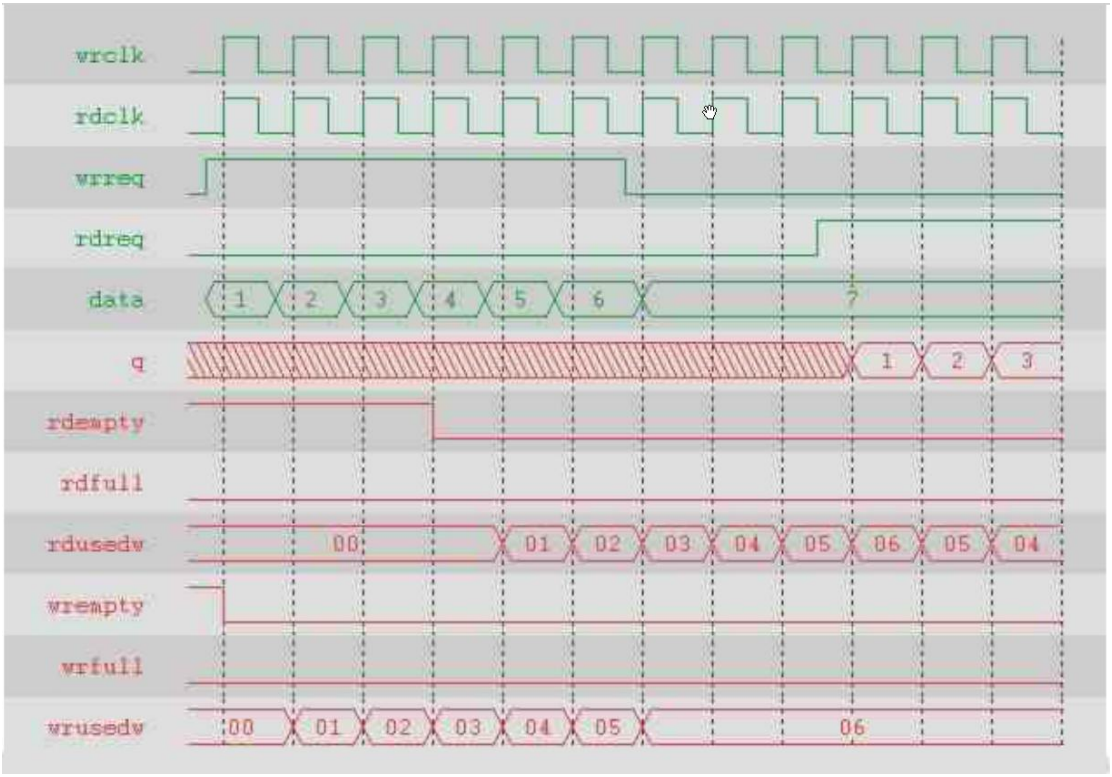
(2) Show-ahead mode(Show-ahead synchronous FIFO mode)

其中：
在 Legacy mode，读端的 rdreq 信号作为读取 FIFO 的请求信号(REQ)，读取数据在 rdreq 置位后的第二个时钟周期有效。

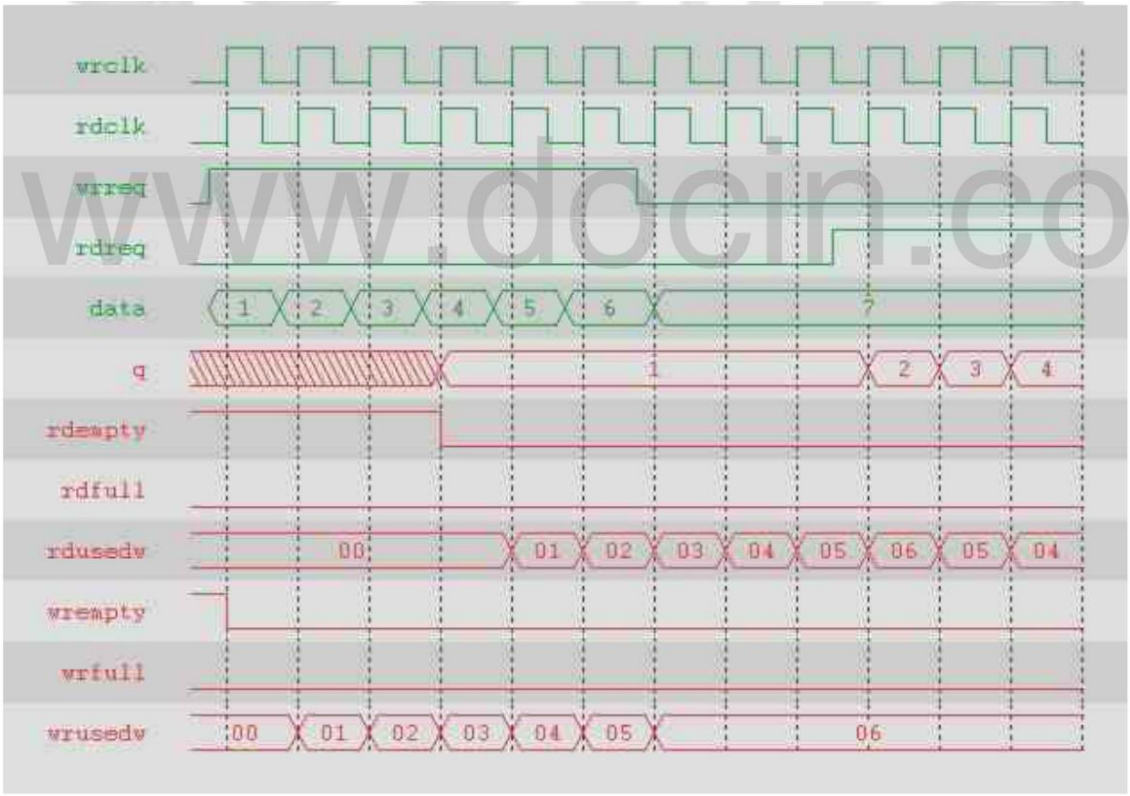
在 Show-ahead mode，读端的 rdreq 信号作为读取 FIFO 的确认信号(ACK)，读取数据在 rdreq 置位后立即有效，不要额外的读取周期。

下面分别给出 Legacy mode 和 Show-ahead mode 的读写时序：

[A] Legacy mode



[B] Show-ahead mode



由上述时序可以看出两种模式的区别。

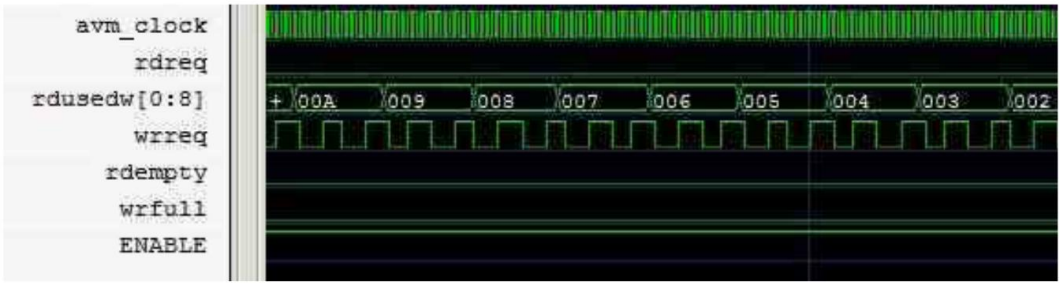
值得注意的是：
读端在读取数据的时候，必须等待写端数据准备好，即 rdempty 为低之后开始读取数据，为高期间表明 FIFO 状态为空，写端写入数据未有效。

相应的在写端如果 wrfull 为高，则表明 FIFO 状态以满，不能再写入数据，此时写入的数据无效。

Altera FIFO 使用注意事项

使用两年多的 Altera 器件，总觉得自己对 FIFO 这类简单的 Core 已经是熟悉不能再熟悉了，到现在的结果是花了三个星期时间解决一个 FIFO 时序问题。

现象

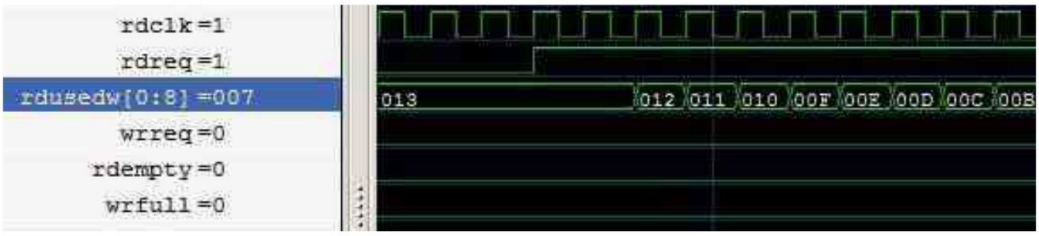


FIFO 实际读写情况

上述图是从 SignalTap 中保存的.vcd 文件，其中 avm_clock 为读时钟，rdreq 在图中都处于无效状态，而写数据一直在进行（wrreq 高电平有效，写时钟未示意），但是 rdusedw 却随着数据写入逐渐减少，到最后直至 wrfull 有效，rdusedw 为零。

分析

Altera 提供的 FIFO Megacore 中有 rdusedw 和 rdempty 两中信号，分别用于描述 FIFO 内的可读数据量和 FIFO 内部为空，但是当你读完 FIFO 内数据的同时，并不会马上响应，即 rdusedw 不会在紧接的时钟为 0，rdempty 不会高电平有效。如果此时，你需要根据 rdempty 信号是否进行下一次读进程，则会发生上述描述的现象。



错误读时序

如上图所示，读完 13 个数据后，FIFO 已经为空。但是 rdusedw 在两个时钟后才为零，由于 FIFO 有读保护逻辑，rdusedw 一直为零。错误已经造成

总结

首先 wind330 希望看过本文的同学不对这种错误费解。然后，尽量不利用 Altera FIFO 的 wrfull 和 rdempty 信号判断是否读写 FIFO，而是利用自己工程中其他信息判断 FIFO 是否为满或空。最后，这种问题无法用行为仿真得出结果