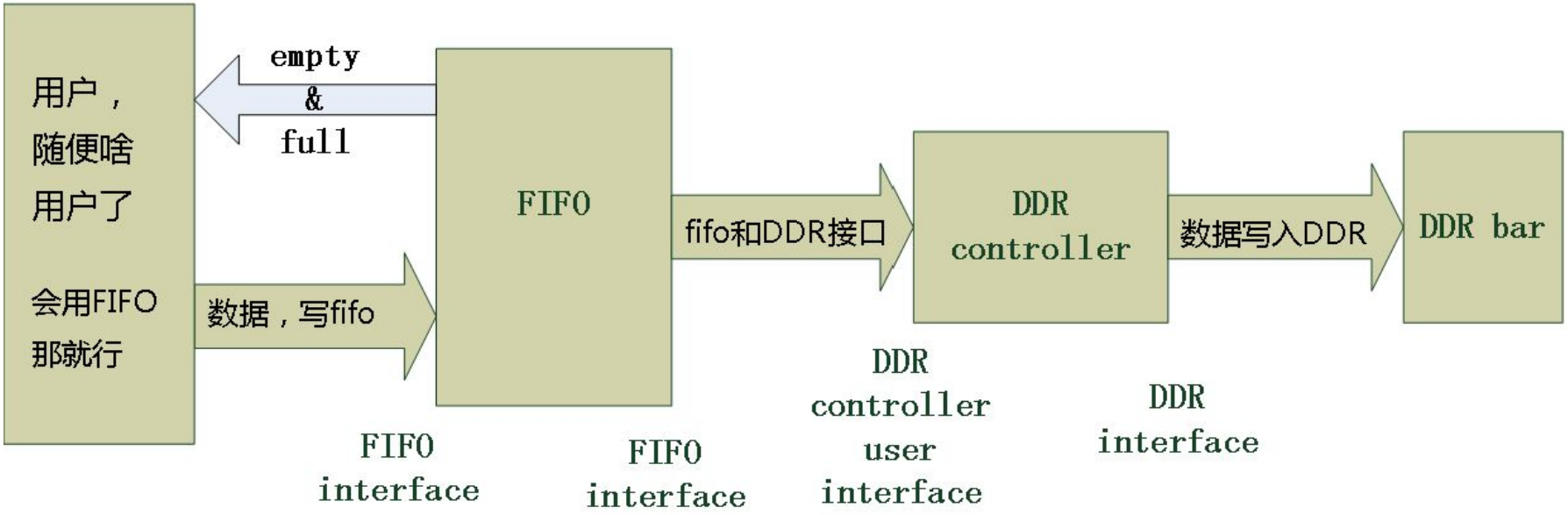
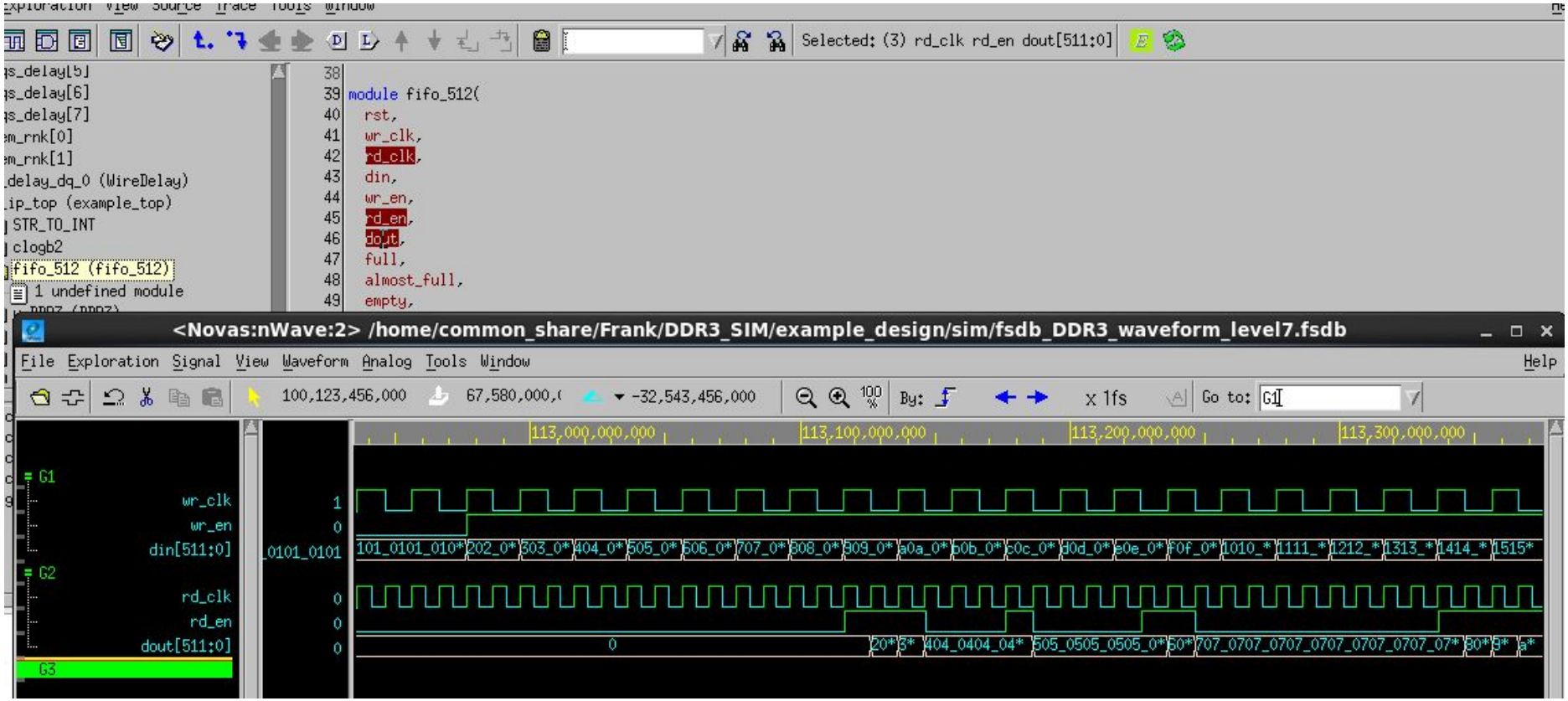
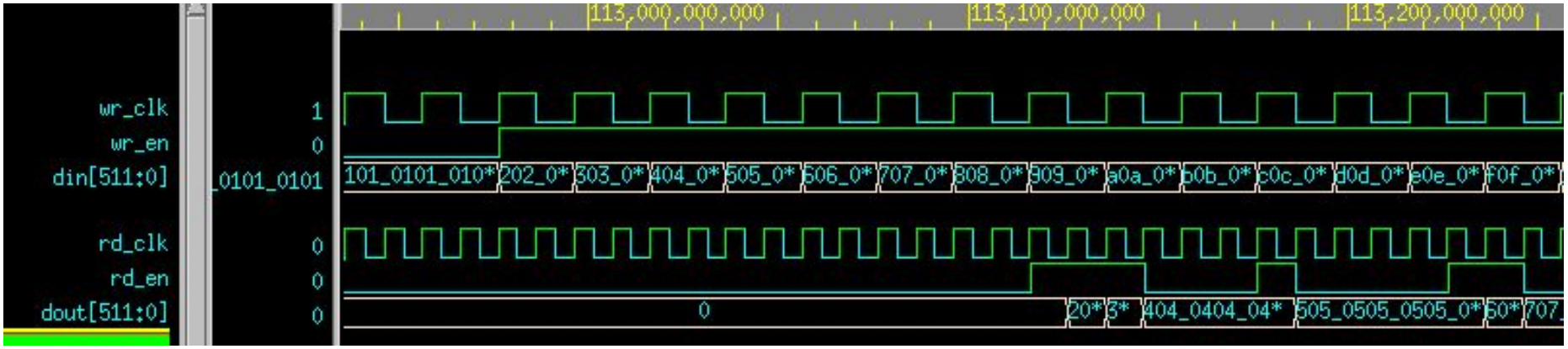
到现在总算可以进入应用部分了
要说应用呢...那就是怎么把FIFO和DDR接起来了。
DDR接FIFO的情况需要考虑FIFO的时序和DDR 用户接口的时序
真等你调出来，那是要花点心思的。
下面我们就来看图说话吧。
至于为什么要接FIFO....
你不接FIFO还想做啥呢？
下面是一个典型的经过FIFO写DDR的应用例子流程图。别看了，很简单的。
废话少说，下一页开始图示FIFO和DDR用户接口的时序。



下面是一个fifo的接口
wr_clk是写时钟，wr_en是写使能，din是写入的数据。
那么rd_clk是读时钟，rd_en是读使能，dout是读出的数据。

发现什么区别了没？
神马？图不够大看不清楚？看下页大图~

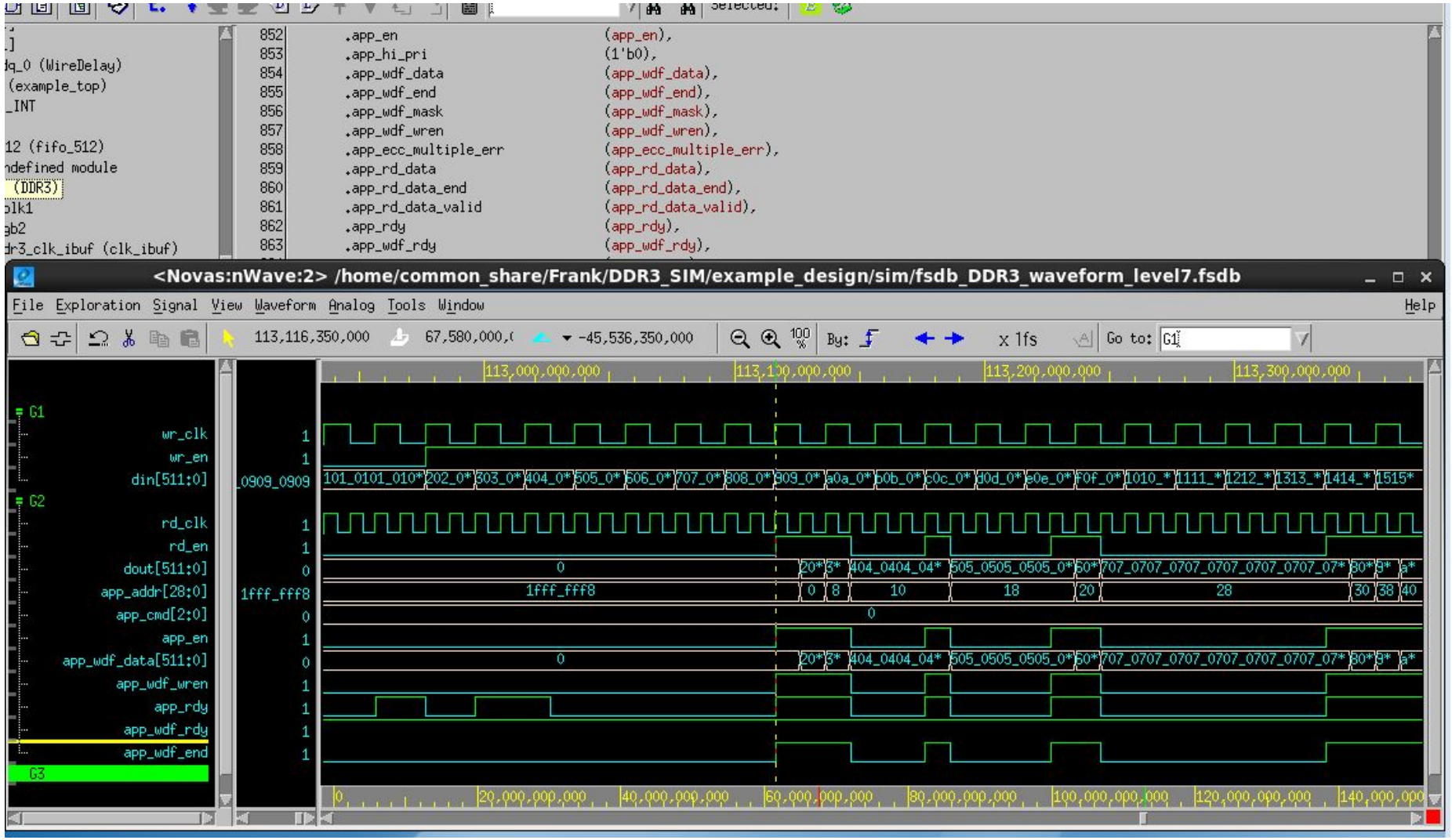


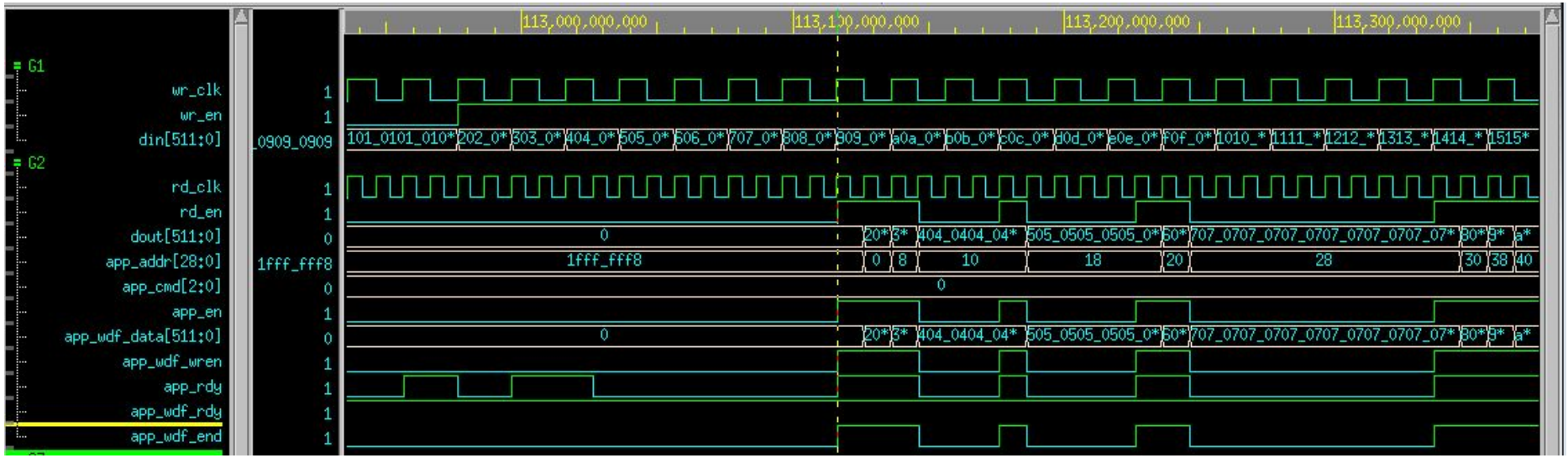



这下图够大了吧？
写入fifo的din在wr_en使能范围内，是2020开始，顺序递增的。
而读出fifo的dout在wr_en使能范围内，第一个数据是初始值，而不是从fifo里读出的第一个数据2020，注意到这一点没有？

那么，如果你希望从DDR条子的0地址开始写入第一个进入FIFO的有效数据，你应该怎么办？

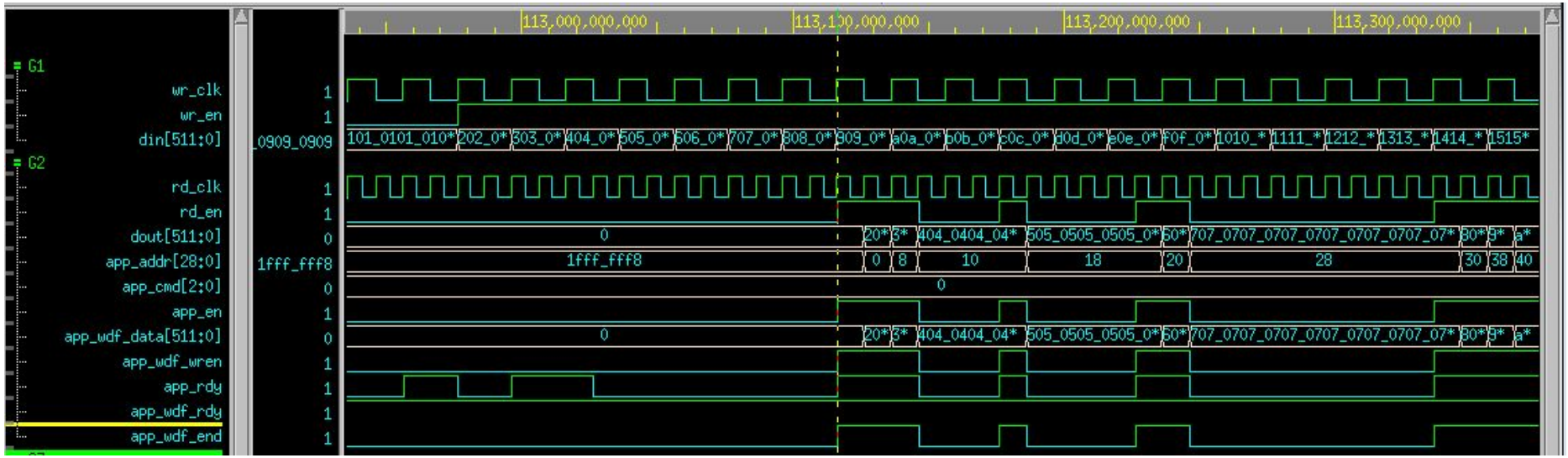
是不是看得脑子有点乱？没关系，我们下一页继续放大了讲。






看到地址了没？
就是app_addr，我们把他的初始值设为1fff_fff8，这是整条4GB的DDR条子的最后一个地址，它再加上29'h8，就又回到首地址29'h0。
如此，地址29'h0写入的值就是第一个进fifo的2020。

等等，这里为啥地址每次都加上29'h8？
你知不知道burst length这一说？
默认的burst length是8，地址自然每次加上8。
不知道突发模式写DDR的，你还是百度一下DDR的基础知识吧...

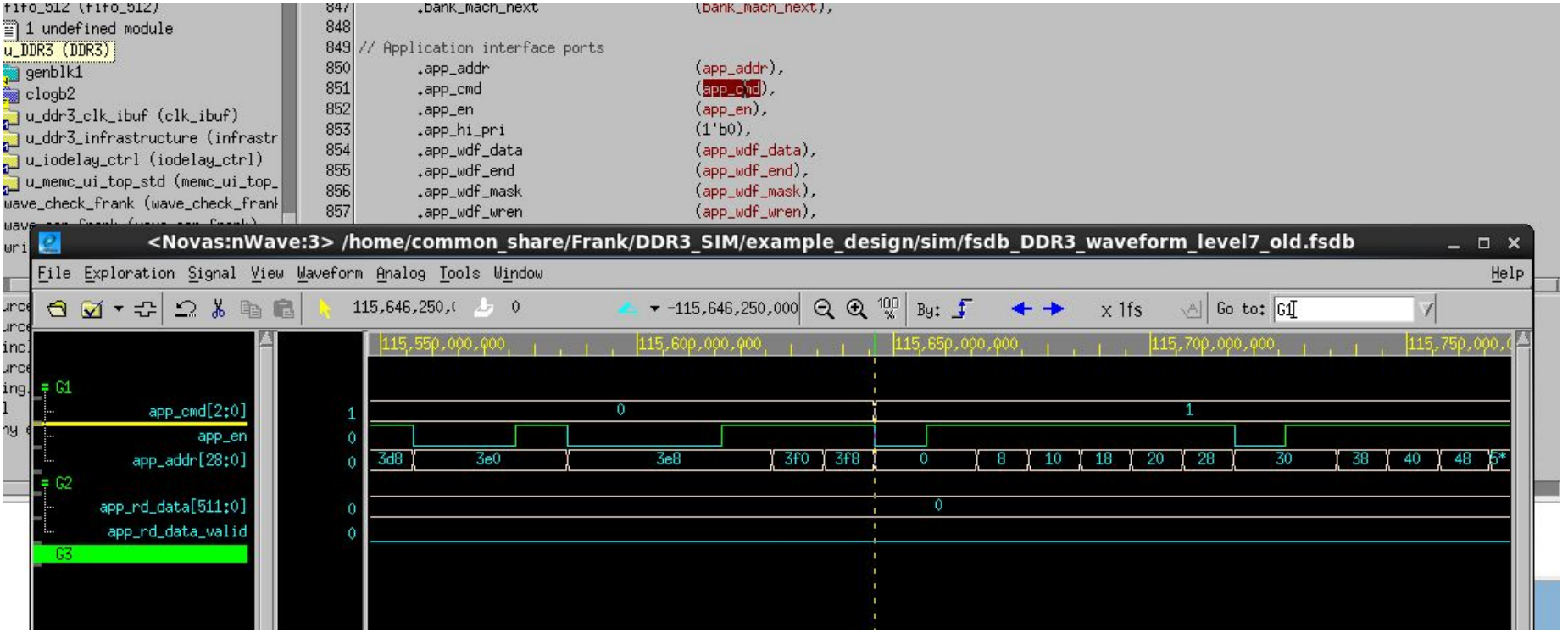



除了地址要打个提前量之外，这个图上值得注意的是各种用户接口信号的时序。

一句话，就是“对齐”
那就是设计教程里讲的，一次成功的写入，需要地址系统和数据系统的各自对齐，并且我这个例子里面，地址系统和数据系统相互之间也是对齐的。

再具体的讲，把能对齐的全都给对齐了，自然就搞定了。

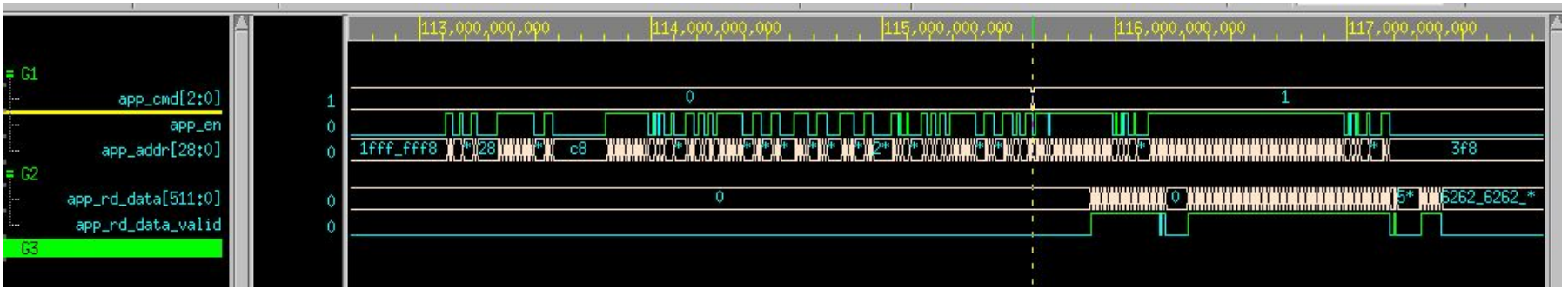
此外，如图所示，app_rdy可以独立拉高，它只表示DDR是准备好了，而你可以不去理它。




至此，写入已经ok
那么怎么读出来？

看上面的图，光标线，注意到图上黄色光标虚线没？什么你还没看到？
那你看到app_cmd的0和1两个值的分界线了吗？

拉高app_en，同时给出你要读的地址。
然后你就等到app_rd_data_valid拉高吧。




到app_rd_data_valid拉高，是要过一阵的。
放大，就是下面的图：

讲一点关于收尾的工作。

刚刚讲的是怎么把开头的数据写对。
事实上如何把最后一个写入FIFO的数据刚好读出，相对麻烦一点。

还有处理fifo被读空，但是后续还有数据会写入的情况，也比较麻烦。

大致的思路是，在fifo可能被读空的时候，发现fifo里还剩一个数据的时候就停止读取（这会用到fifo的almost_empty信号）。

这样fifo被读空的时候，最后一个数据会留在数据总线上，等到下次激发写DDR功能的时候，第一个数据就直接被打入DDR。

对最后一个数据，需要一个独立的处理机制来处理，确认fifo被读空之后才停止读取fifo和写入DDR。

这部分内容相对复杂，但是我相信你只要花时间就一定可以调出来。
我是写了一个8个状态的状态机来实现的。

每个人的应用需求不一样，代码还是各人写各人的吧。

好了，DDR的四部教程先写到这里了。