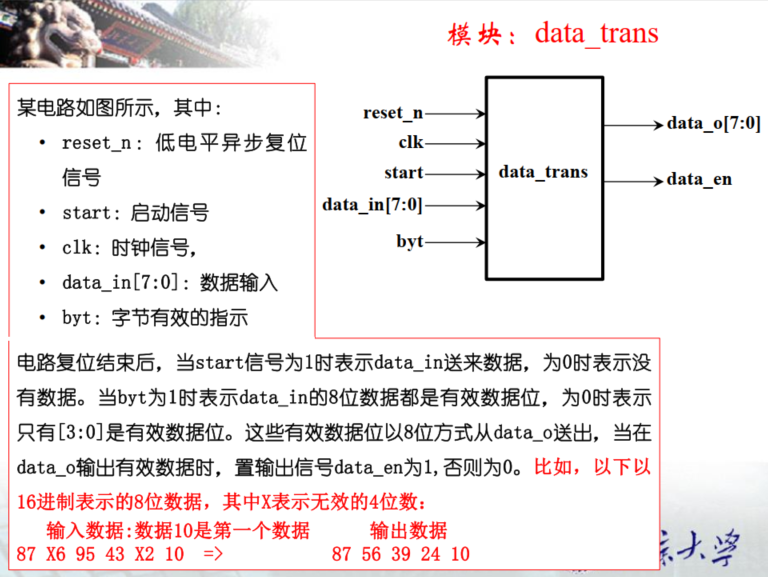
**数据转换器作业**

**模块1——data\_trans设计**

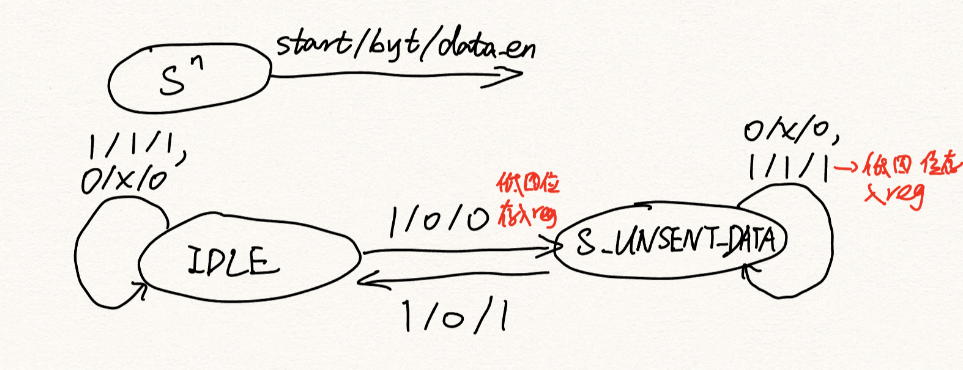
**目标分析**



**设计思路总结**

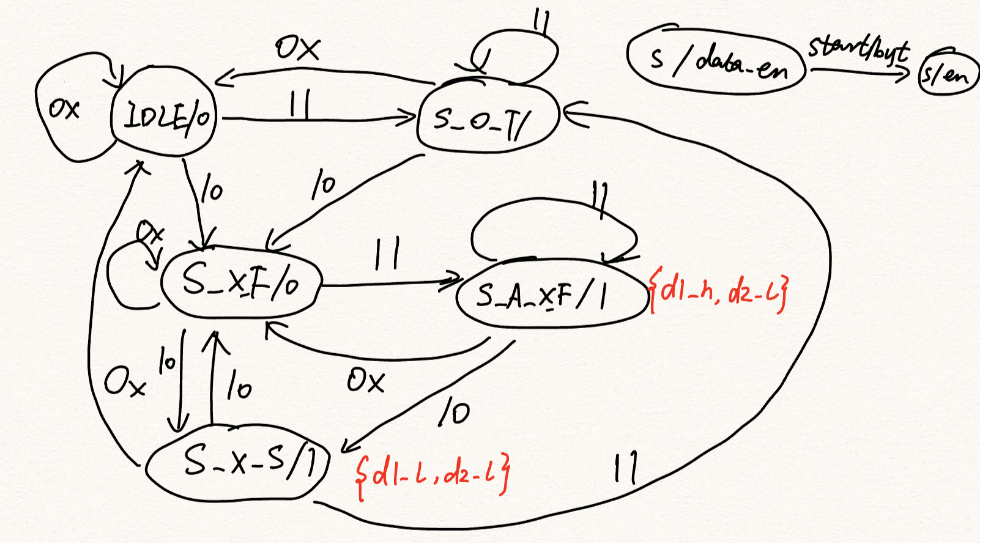
这个模块有多种实现方法。例如：

* 本次作业采用的方法——米里型状态机方法，具有1个clk的延迟，需要2个状态，状态数量最少，设置的reg最少为4位。状态转换图如下图所示：

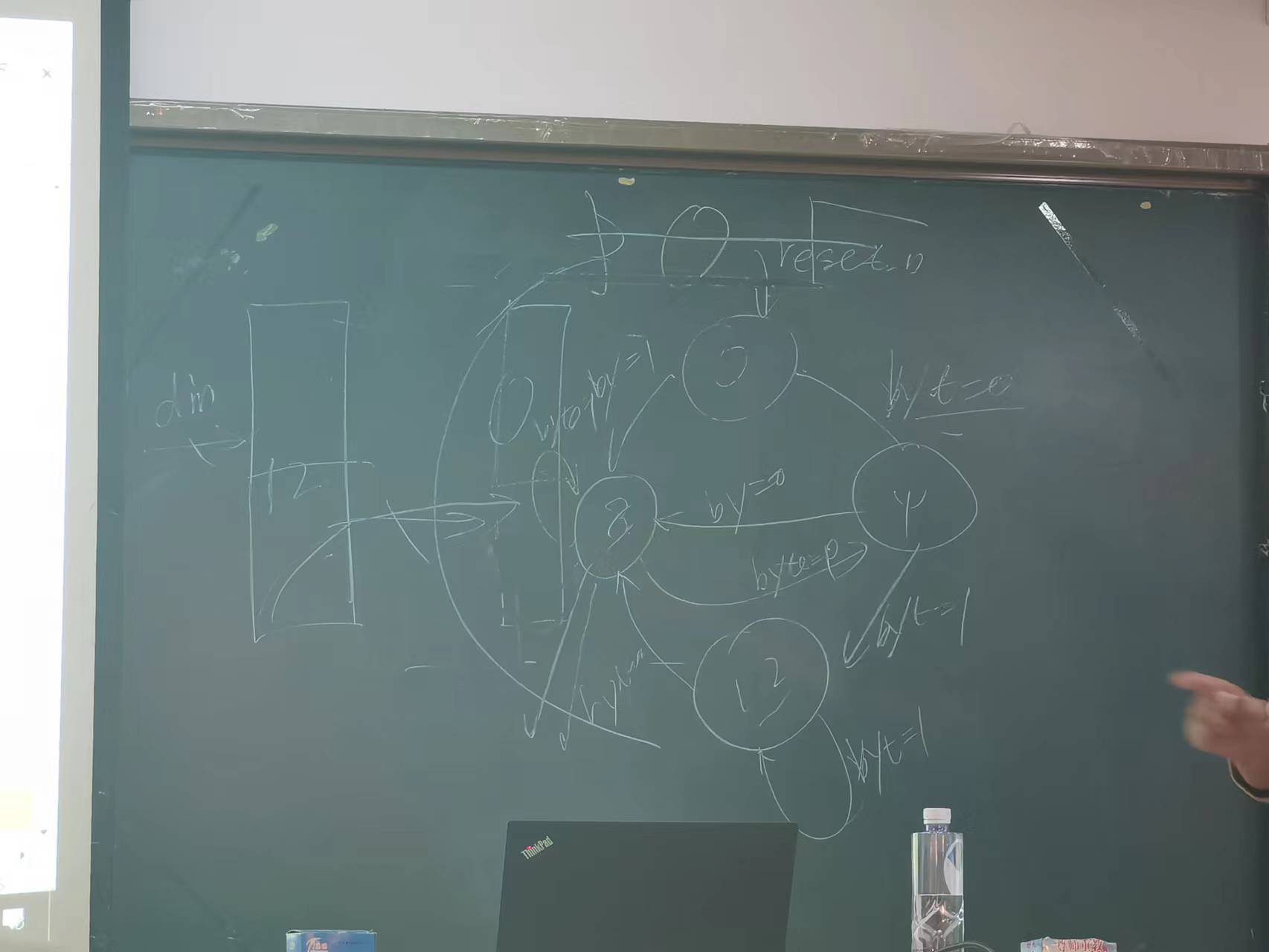


* 摩尔型状态机方法，分为两种情况

1. 1个clk延迟，具有5个状态，状态转换图如下图所示

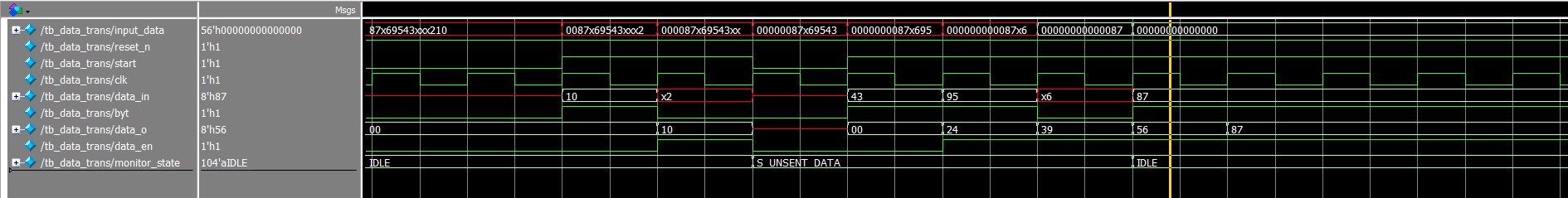


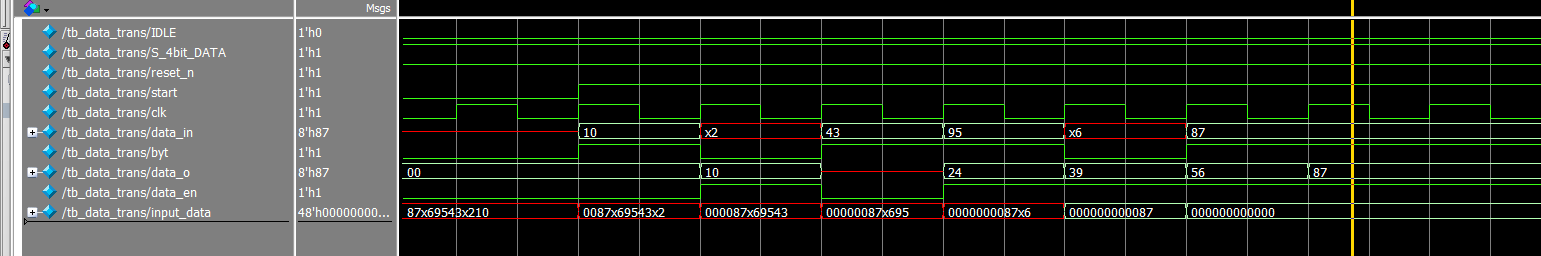
1. 2个clk延迟，需要4个状态，状态转换图如下图所示：



**Testbench验证**

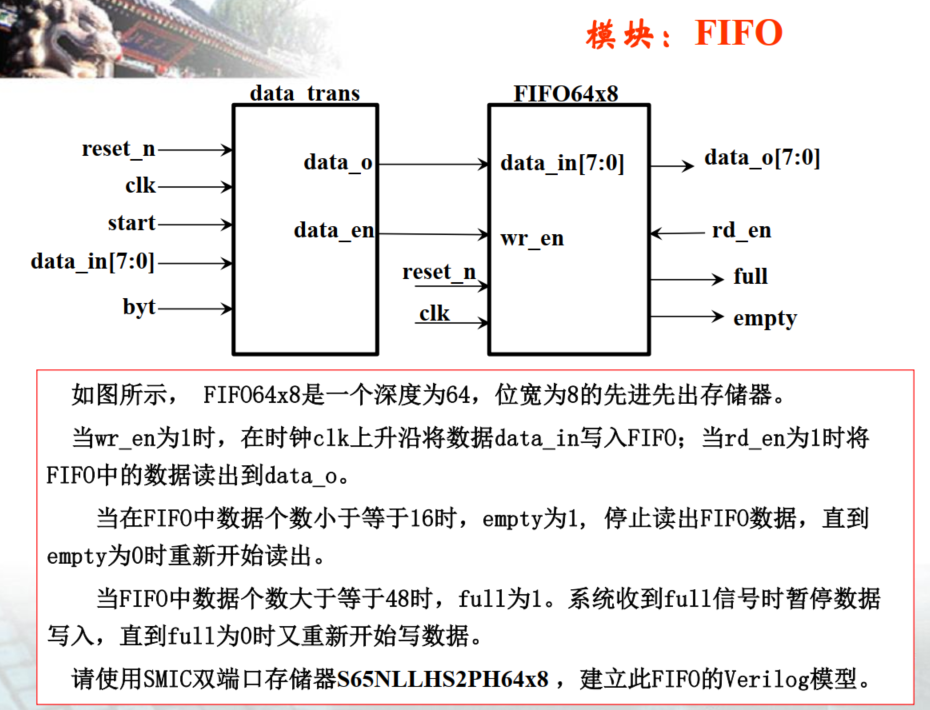
欧：





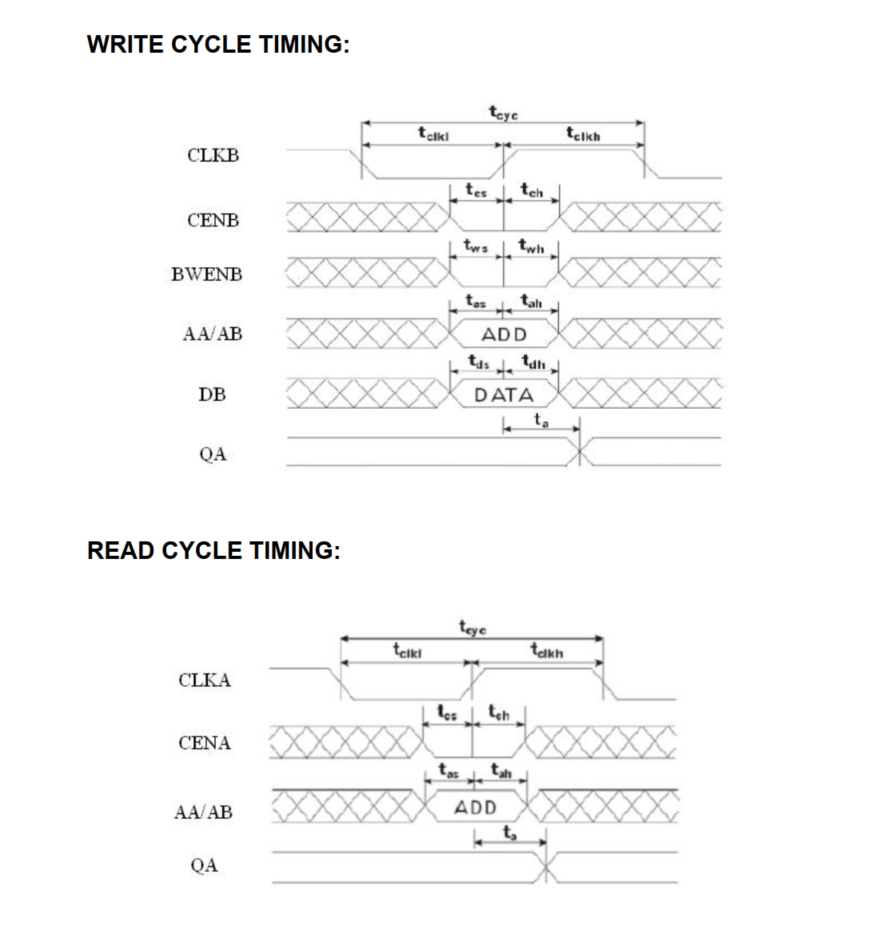
**模块2——FIFO设计**

**目标分析**



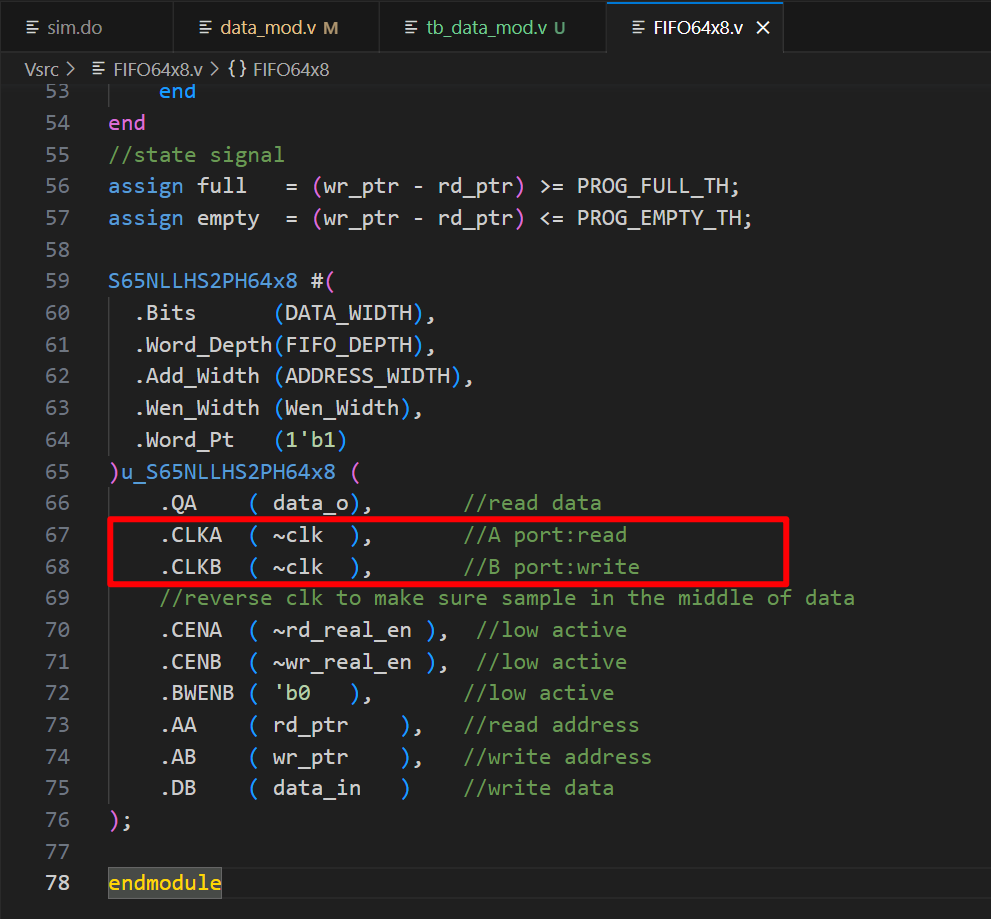
**设计思路总结**

**FIFO读和写的时序要求、采取的仿真策略和行为分析**



根据双口RAM(TPRAM,two-port RAM) S65NLLHS2PH64x8 的数据手册要求，在进行时钟读/写操作的时钟上升沿时，要求使能信号、地址、数据等输入信号满足一定的建立时间和保持时间，否则如果不满足的话，根据仿真测试结果，该器件无法进行正常读/写操作。

为了让S65NLLHS2PH64x8正常工作，一种方法是给它的输入信号增加延迟，例如让使能信号、地址、数据等输入信号晚半个时钟周期到达，保证采样时满足建立时间和保持时间；另一种方法是，让它的时钟信号增加延迟，而输入信号到达时间不变，这种思路下简单的做法是，将时钟信号取反（增加180%的相位，等同延迟了半个时钟周期）送入到S65NLLHS2PH64x8的CLKA（读时钟信号）和CLKB（写时钟信号）端，这种方法同样达到了在输入信号变化前后的正中间采样的效果，可以满足正确仿真。本次作业采用了方法二，将时钟信号取反的办法。



现在分析将时钟信号取反后送入到S65NLLHS2PH64x8之后，FIFO的行为是什么样的？FIFO的读和写到底是会快了半个时钟周期还是慢了半个时钟周期？

因为内部的S65NLLHS2PH64x8器件使用的读和写时钟与外部时钟相位差了180°，原本FIFO在检测到读和写信号的第一个时钟上升沿进行读或者写数据的操作行为，而现在相位差了180°后，FIFO在检测到读和写信号的第一个下降沿就会进行读、写操作了。因此读会比一般FIFO快半个时钟周期产生想要读的数据data\_o，同样，写也会比一般FIFO快半个时钟周期写入数据。因此结论是，FIFO的读出数据的产生，以及写入数据到存储中两种行为，都快了半个时钟周期。

**空和满信号的产生思路**

DesignWare的三种空：全空，接近空（almost\_empty，带门限），空一半（half\_empty）

fifo的空满控制，一种方法是减法器，另一种实现是设置cnt代表存储器中的数据个数，每次随着读写更新cnt。

本题目中，由于产生的full信号实际是设定有阈值的几乎满信号，并且该阈值不为fifo最大深度，那么几乎满和几乎空信号的产生，可以直接用位宽为ram地址宽度的wr\_ptr和rd\_ptr相减产生，理由如下：当wr\_ptr>=rd\_ptr时，wr\_ptr-rd\_ptr即fifo中现存的有效数据个数，不用多解释，当wr\_ptr<rd\_ptr时，wr\_ptr-rd\_ptr=fifo\_depth-rd\_ptr+wr\_ptr，结果仍然为现存的有效数据个数。

*可以举个例子验证，假设fifo\_depth=16，wr\_ptr=1010，rd\_ptr=1100，当 waddr<raddr 时，计算两者的差值为16 – rd\_ptr+ wr\_ptr→ 10000 - 1100 +1010 = 1110，此时的 waddr – raddr → 1010-1100 →1010+0011+0001=1110，两者结果相同。*

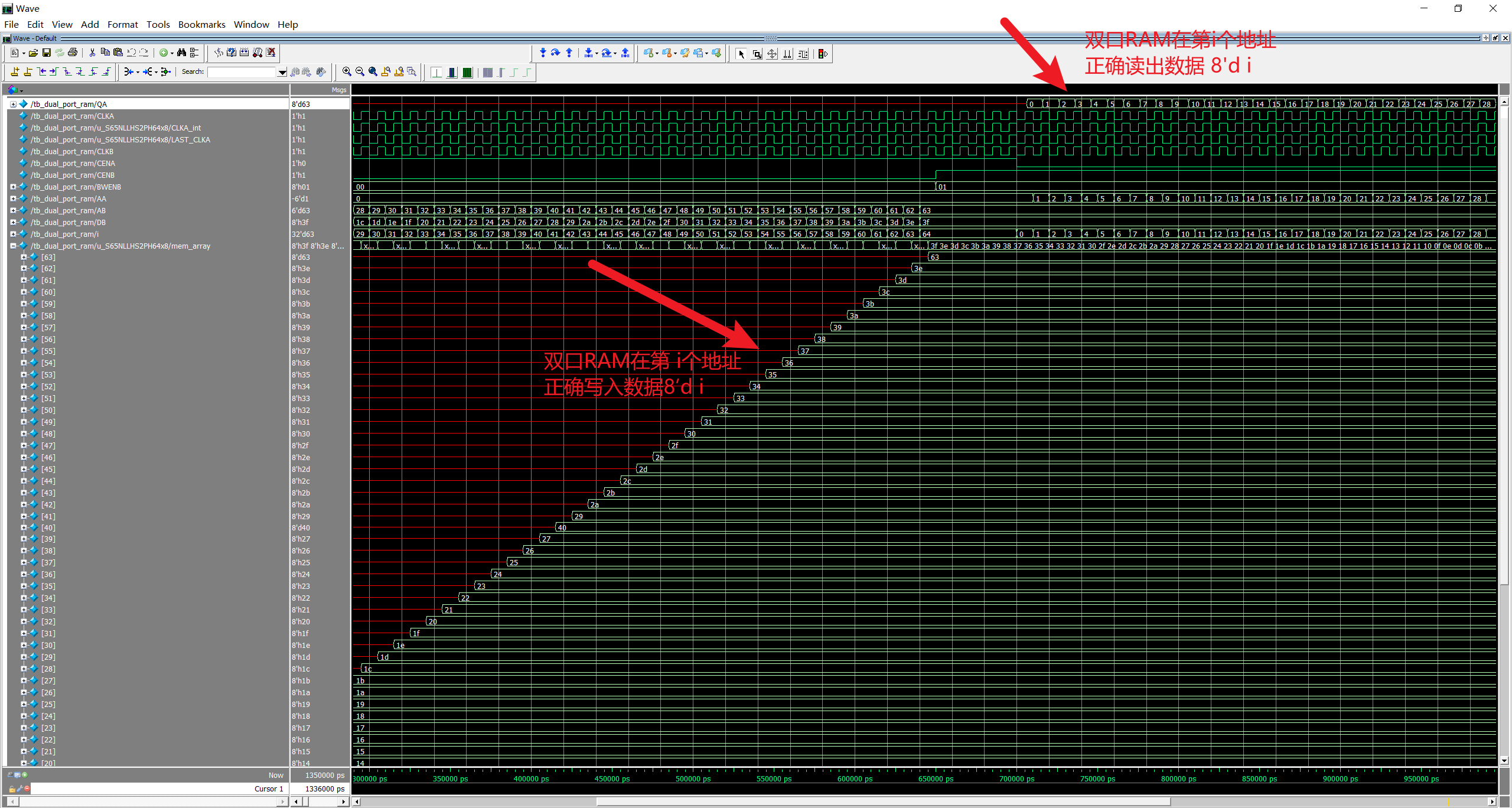
但需要注意的是，本题目中，因为几乎满信号阈值不为fifo最大深度的fifo，所以认为fifo不会出现完全写满的情况，因此wr\_ptr=rd\_ptr时，wr\_ptr-rd\_ptr即fifo中现存的有效数据个数。如果fifo设置了全满信号full，在fifo可以完全写满的情况下，就存在两种情况使得wr\_ptr=rd\_ptr，分别是全空和全满，在这个时候空、满标志的判断就不能简单认为了，一种解决办法是让wr\_ptr和rd\_ptr增加一位位宽，这种方法实现原理可在该文章的分析找到：https://blog.csdn.net/HouQi02/article/details/51683635。

**Testbench验证**

**ram读写测试**

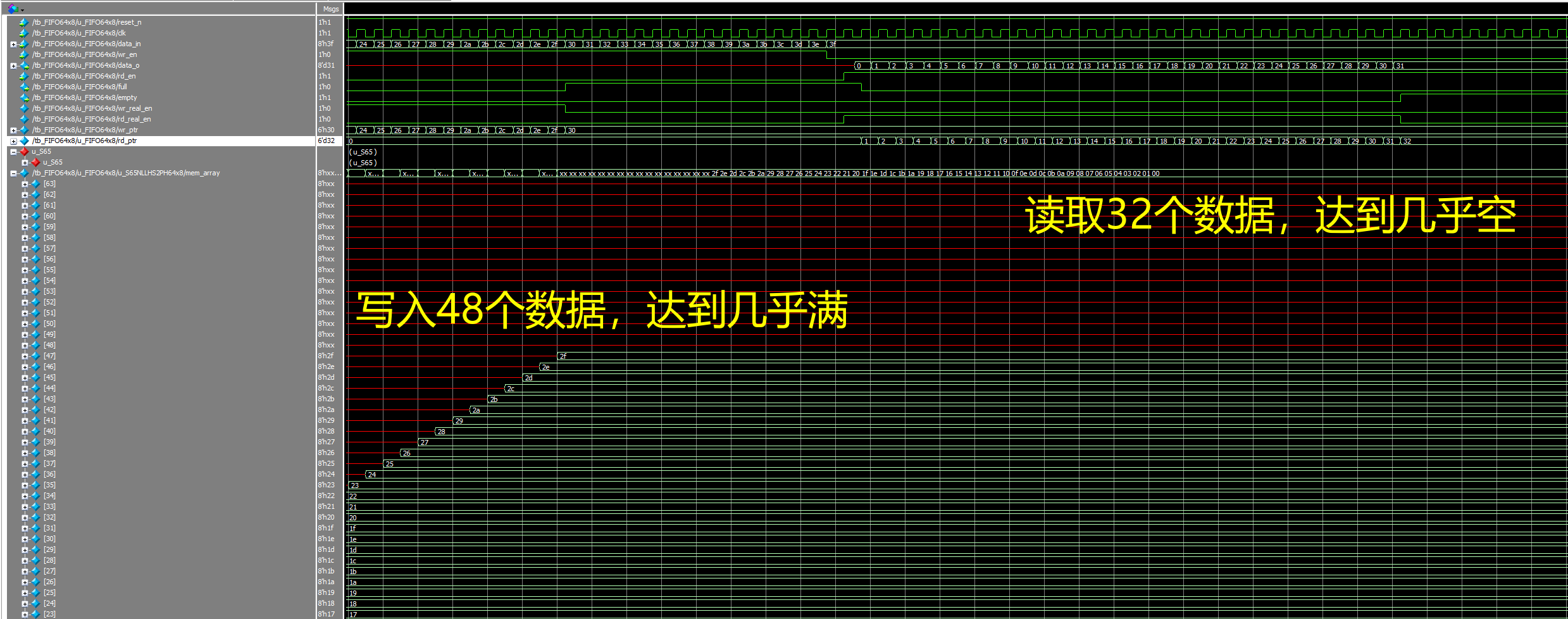
该模块时序检查很严格，如果不满足建立时间和保持时间，那么就无法完成读/写操作。

安装上述分析，对ram读写单独进行测试。



**FIFO读写测试**

在测试完ram后，测试设计的FIFO的功能





mod：

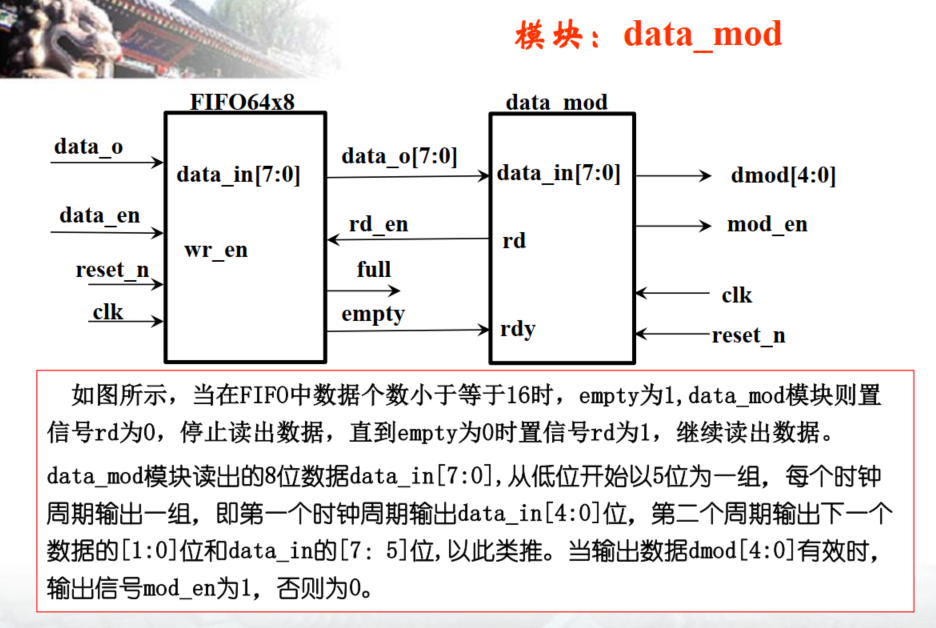
1. 摩尔型状态机：8个状态
2. 移位寄存器实现：

简化输出逻辑，每次只输出固定位置的5位

一眼看不清楚一个always块的作用，代表写的复杂了，要拆开

**模块3——data\_mod设计**

**设计目标**



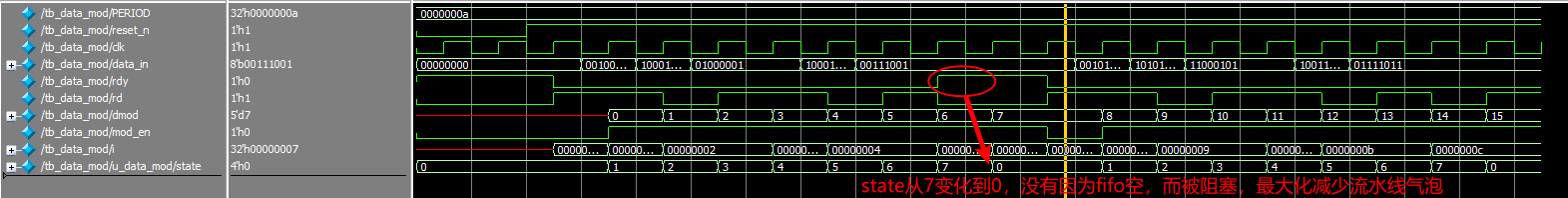
**设计思路总结**

这个模块利用状态机进行设计，reg的宽度最小可以为7位，作业设置为8位，使得逻辑较为简便。

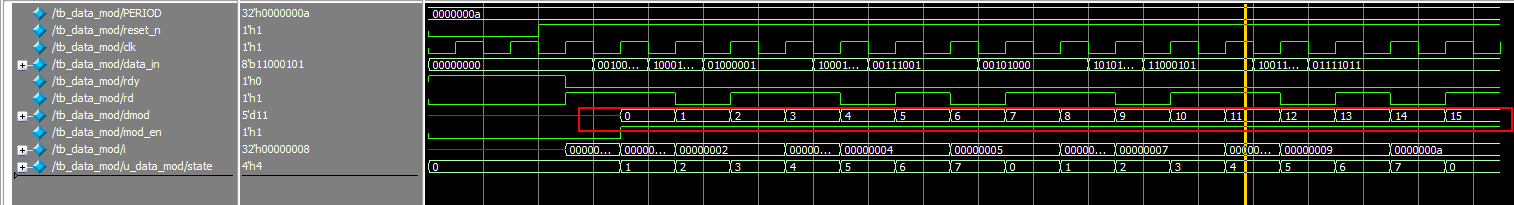
状态机需要8个状态，因为这8个状态是线性变化的，所以在敏捷开发的条件下，为了压缩代码量，可以用一个counter代表状态，代码采用的是这种方法。

在状态为2或者5或者7时，不需要读取FIFO中的数据，可以直接输出reg当中的数据，所以为了最大化压缩流水线气泡，设置了状态机变化条件不是只决定于rdy（empty）输入，而是决定于rd || state == 2 || state == 5 || state == 7。

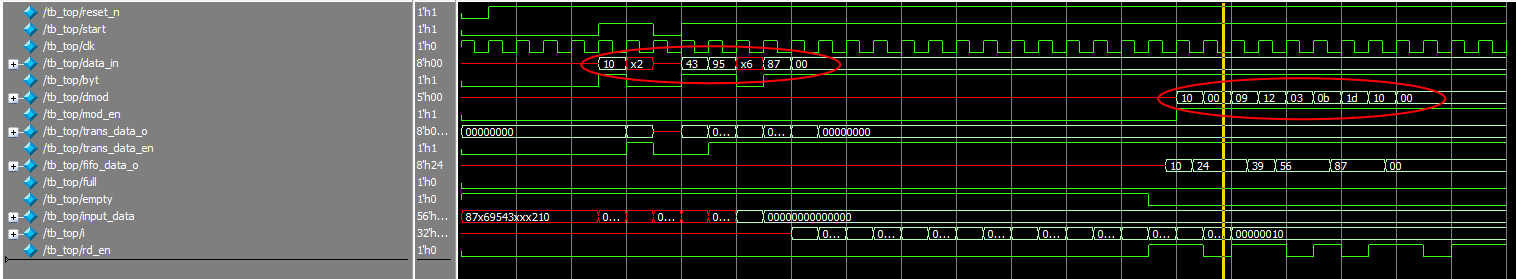
**Testbench验证**

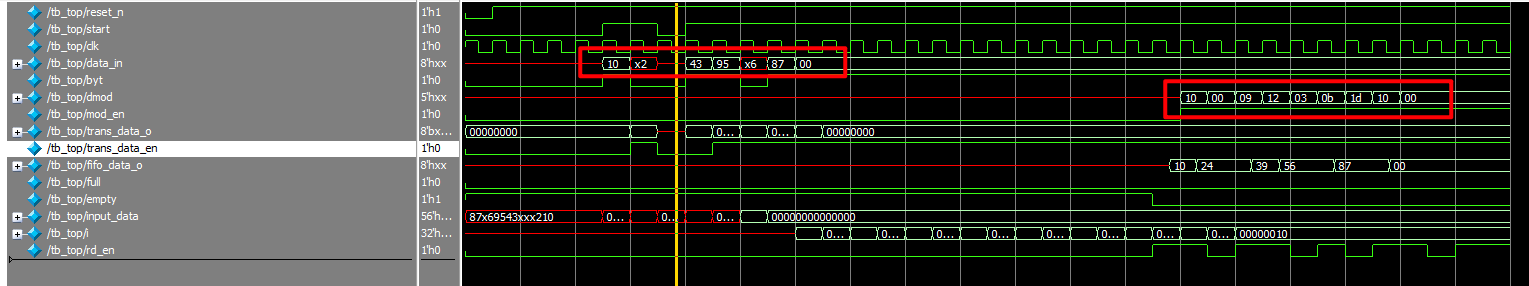


欧：



**TOP模块联合调试**





前两个输入数据为8'h10和8'h43，分别为“8’b00010000”和“8’b00100100”，对应的mod模块输出结果应该依次为5’b10000，5’b00000，5’b01001，即5‘h10，5‘h00，5’b09，可以看到前三个输出是正确的。验证后面的数据，也都是正确的。因此模块功能正确。

**其它思考**

本题目，在异步情况下，即data\_mod和fifo的读时钟clk\_read如果和data\_trans和fifo的写时钟clk\_write频率比值大于8/5=1.6，就可以实现整个系统的读速度大于等于写速度。