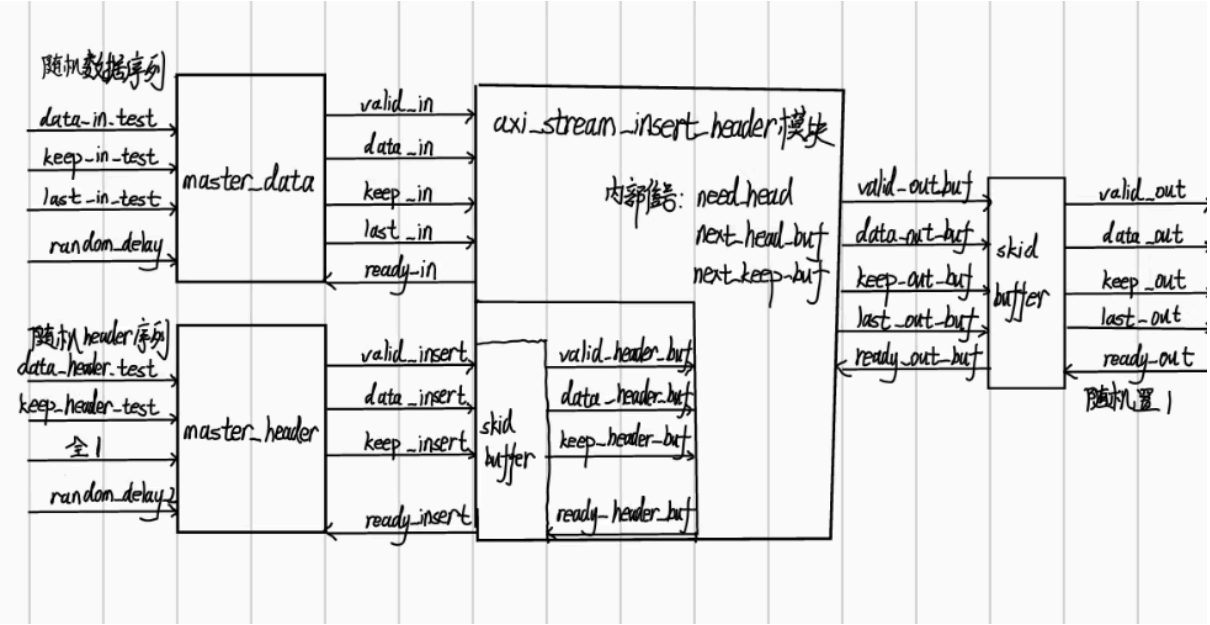
simulation report

1.仿真架构

1.1 data和header输入端

由于数据输入端需要满足AXI stream协议要求，因此不能直接向待测模块提供激励。为此我引入了自己实现的AXI stream master作为中介，首先向master提供随机生成的data和header序列，再由master向待测模块发起请求，结构图如下



1.2 data输出端

输出端只需要提供ready_out的激励信号，采用随机置1的方式即可模拟最恶劣的情况

2.激励设计

```
parameter HEADER_NUM = 2;
parameter AVG_LENGTH = 3;
parameter DATA_NUM = HEADER_NUM*AVG_LENGTH;
parameter NO_INTERRUPT = 0;
```

2.1 header输入

预先生成共HEADER_NUM个header，每个header的内容随机，使能信号keep随机。

NO_INTERRUPT==0时，master_header会在每次header发送之间停顿随机时间；

NO_INTERRUPT==1时，master_header会连续发送header

2.2 data输入

预先生成共HEADER_NUM次请求，每次请求的数据帧数量随机，但是总数为DATA_NUM，每个数据帧的内容随机，最后一帧的使能信号keep随机。

每次请求的过程中可能随机中断然后继续请求。

NO_INTERRUPT==0时，master_data会在每次请求之间停顿随机时间；

NO_INTERRUPT==1时，master_data会连续请求

> data和header的请求彼此独立

2.3 输出端ready_out

采用随机置1的方式模拟slave端

3.仿真结果

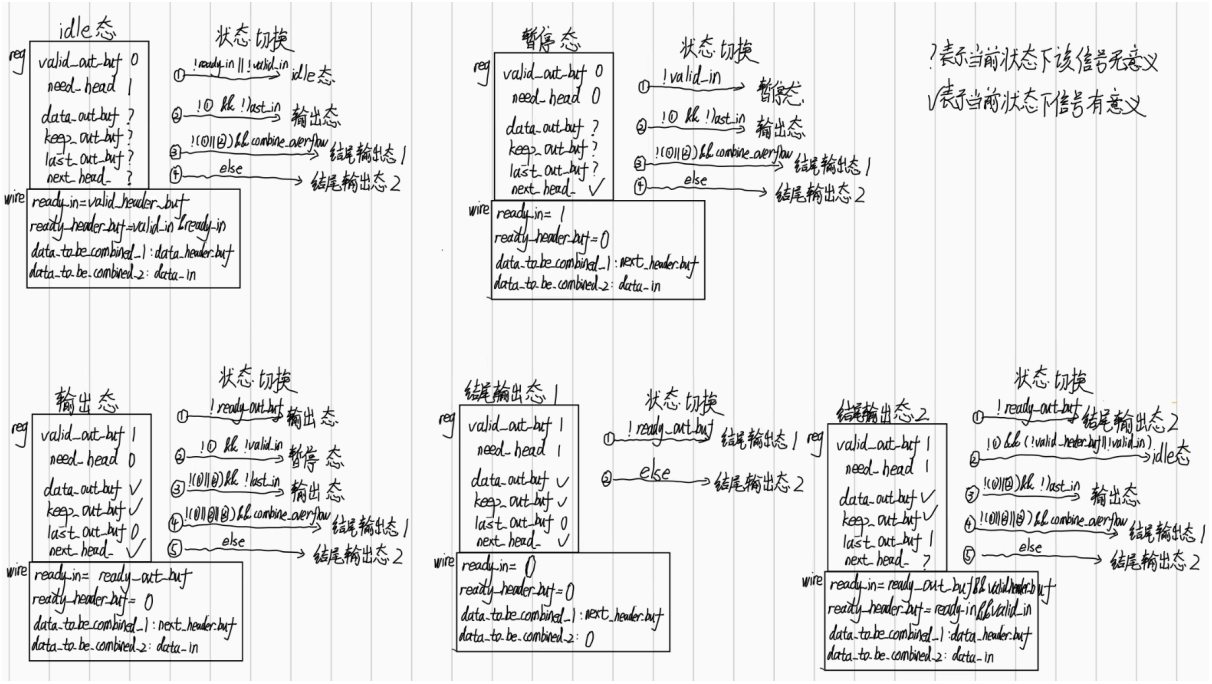
3.1状态图

虽然实际实现未采用状态机方式，但是模块工作过程仍等效于一个状态机，如果不同状态切换过程的波形图符合预期，则可证明模块的正确性。

该模块共有五种状态，分别为：

```
idle态：当前并未进行输出，等待data和header准备好
输出态：当前正在输出一个数据流，并且对应数据流的输入也在进行中
暂停态：由于数据流的输入意外中断，暂停输出并等待输入恢复
结尾输出态：数据流的输入已到达最后一个数据帧，此时需要等待该数据流输出完毕
    重新接收新的数据流。由于header的存在，剩余输出部分可能需要两次请求才能
    因此该状态细分为1和2两个状态
```

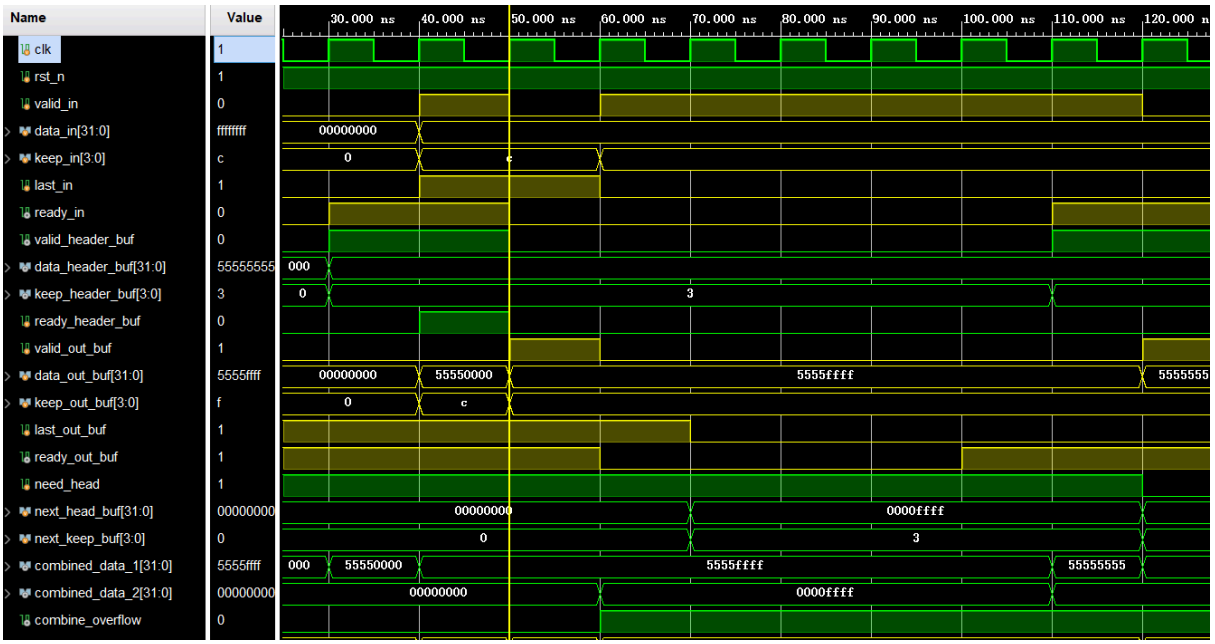
各状态的内部信号情况与状态转移情况如下：



3.2 状态切换波形图（以DATA_WD=32为例）

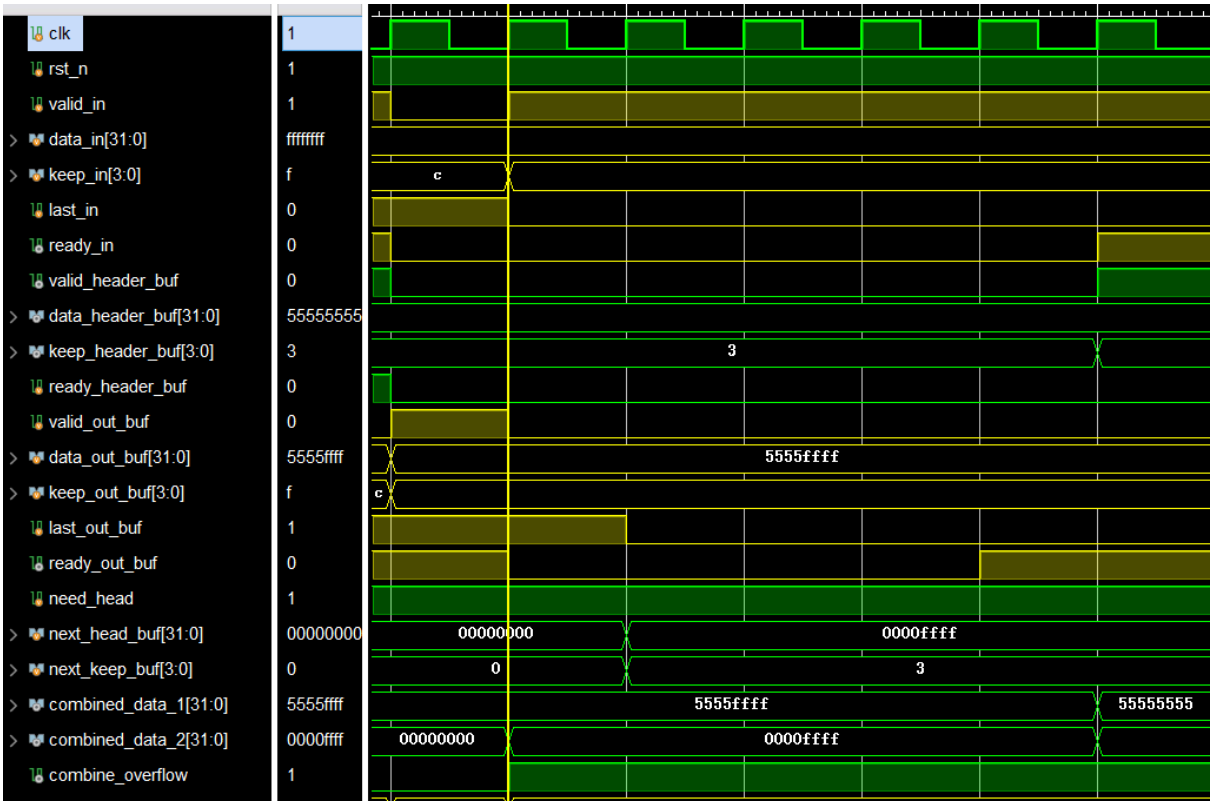
3.2.1 idle态→结尾输出态2

黄线左侧周期，header和data均已到达，由于header仅半字有效，data已经是最后一帧且仅半字有效，二者拼接后可一次性输出，进入结尾输出态2（黄线右侧周期），data_out_buf为0x5555ffff正确，恰好此时ready_out_buf为1，输出完成



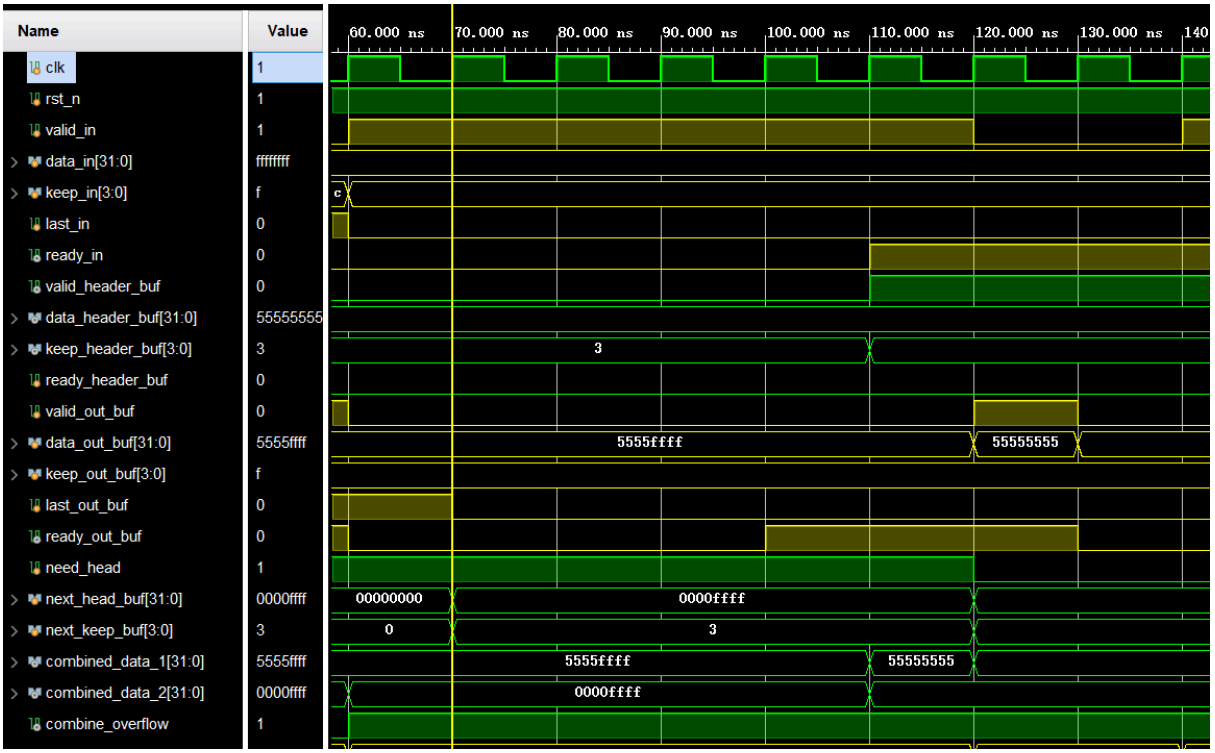
3.2.2 结尾输出态2→idle态

黄线左侧周期完成了上一个数据流的输出，此时没有新的数据流输入也没有新的header输入，因此回到idle态



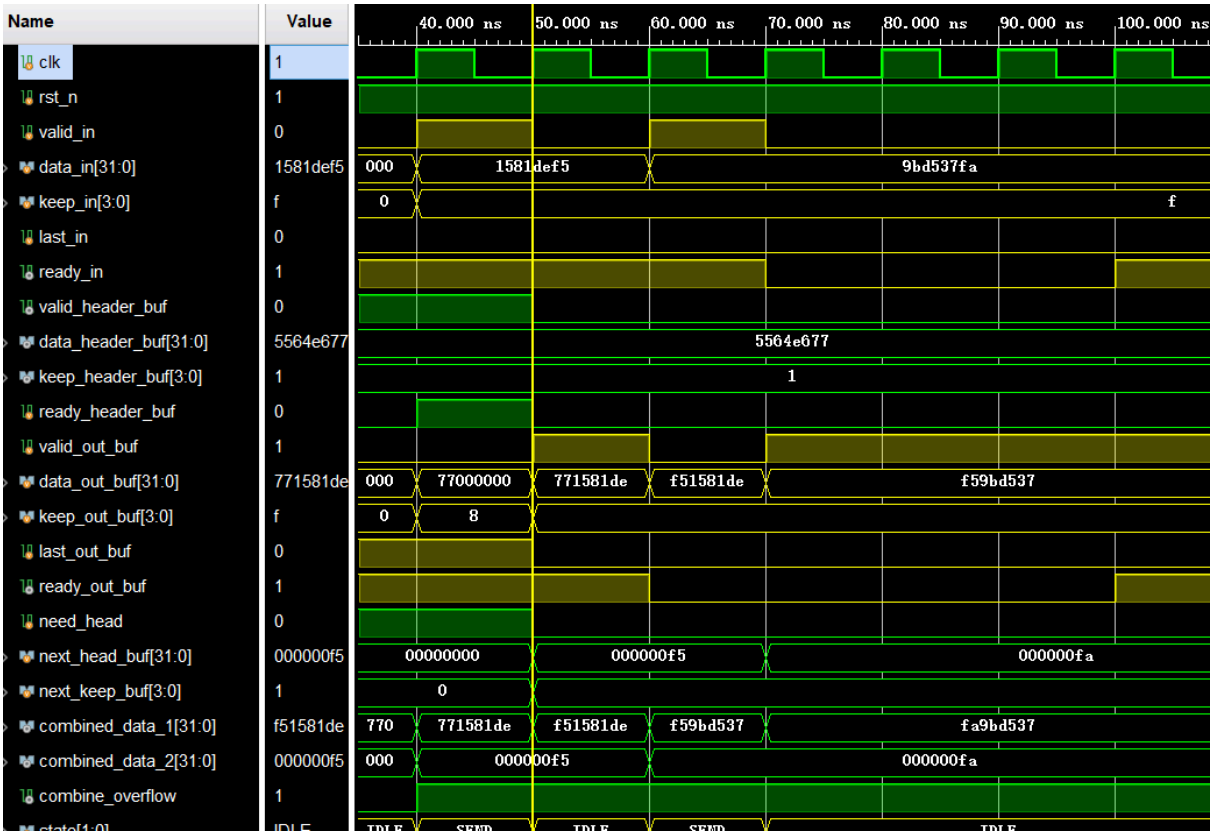
3.2.3idle态→idle态

黄线左侧周期处于idle态，此时有新的data_in到来，但由于header尚未到来，因此置ready_in为0阻塞输入数据流，保持在idle态



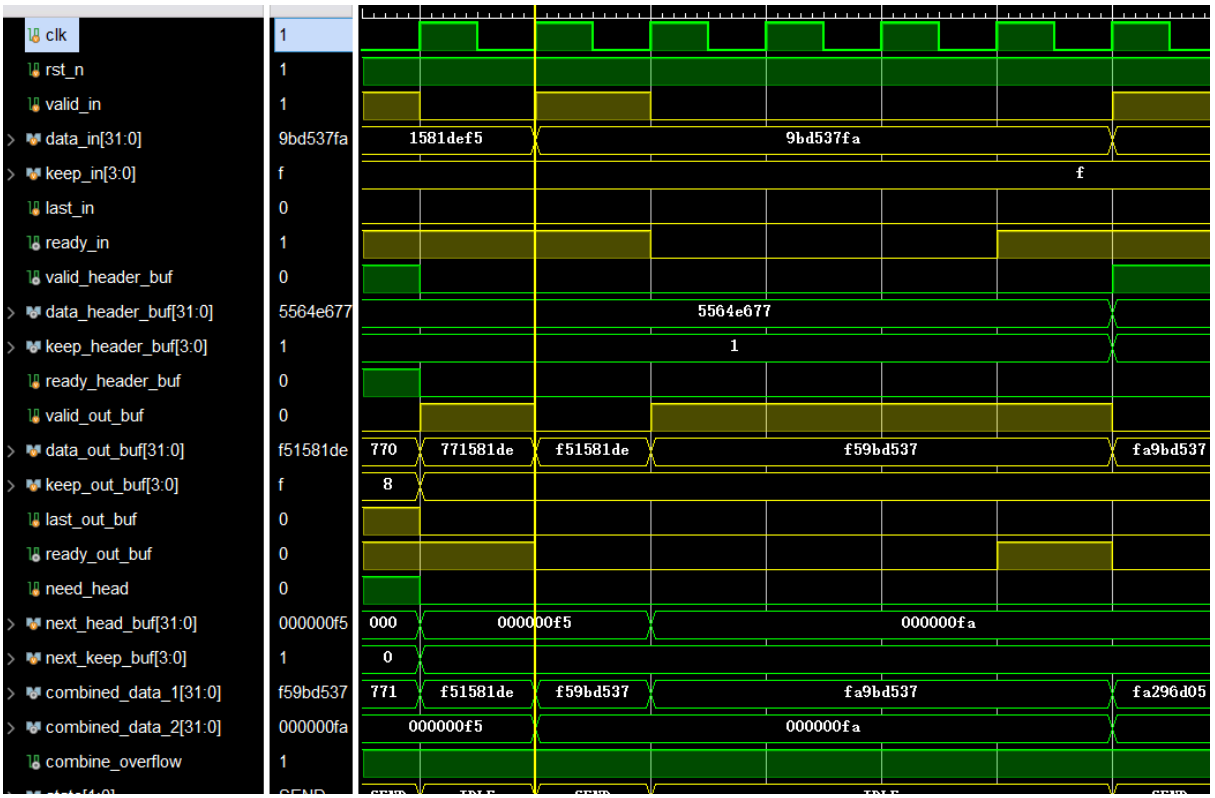
3.2.4 idle态→输出态

黄线左侧周期header和data均到达，且data并不是最后一帧，因此进入输出态，data_out_buf为0x771581de正确



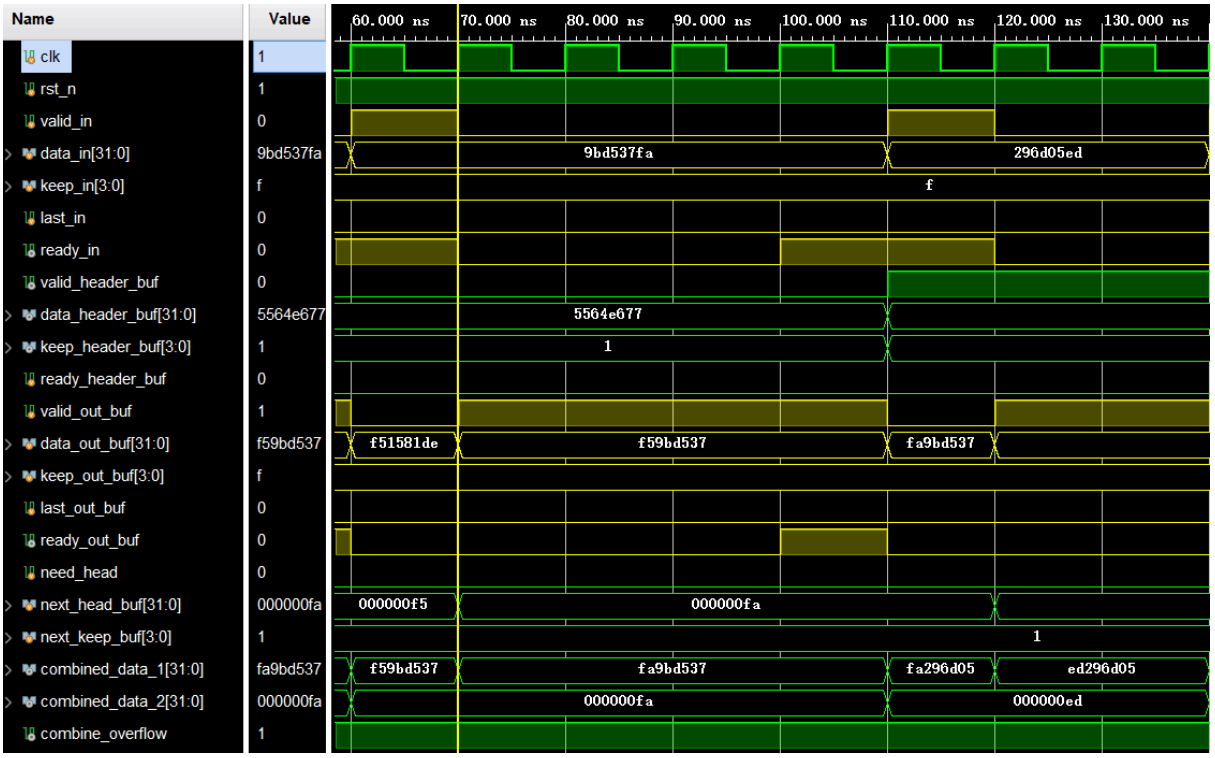
3.2.5输出态→暂停态

黄线左侧周期处于输出态，但由于data_in数据流暂时中断，进入暂停态，valid_out_buf为0暂停输出，等待输入恢复



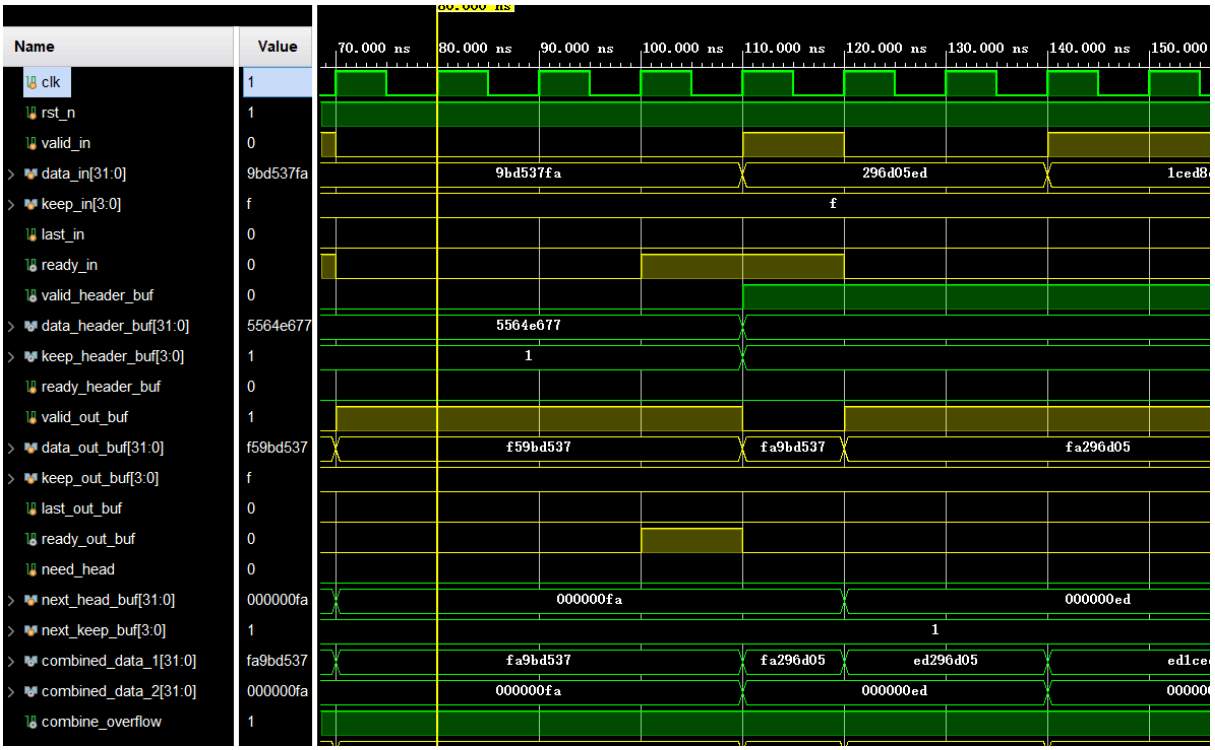
3.2.6暂停态→输出态

输入恢复，且data_in未到达最后一帧，进入输出态，data_out_buf为0xf59bd537符合预期



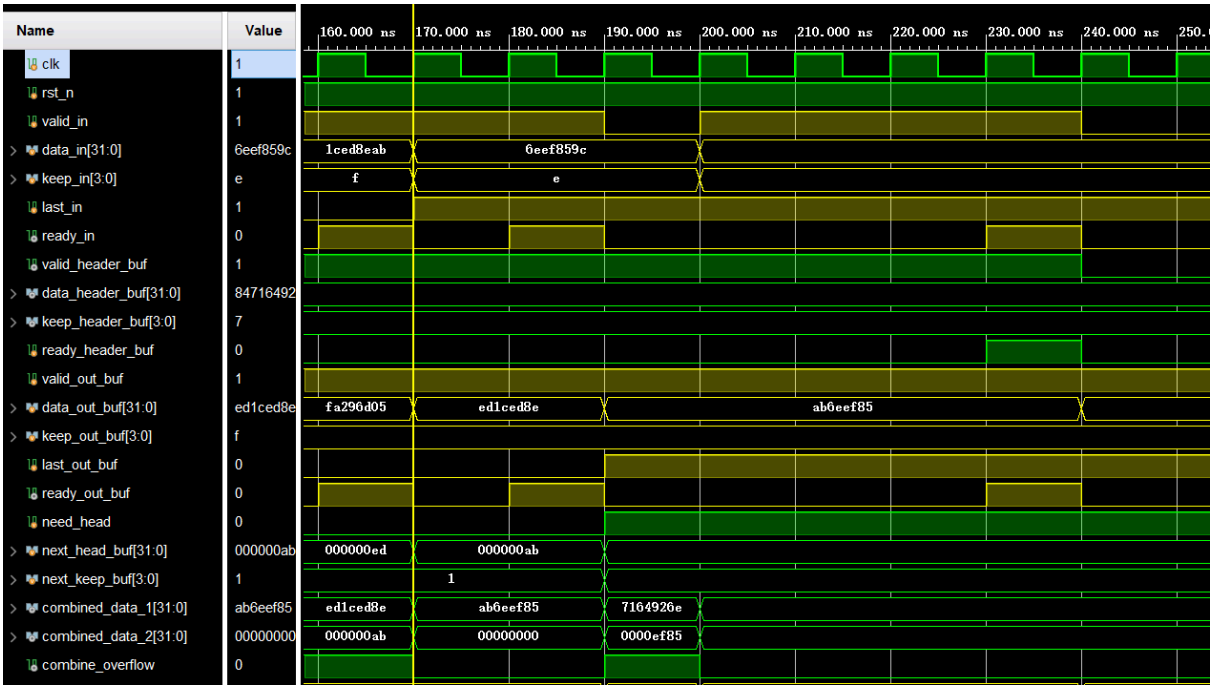
3.2.7输出态→输出态

由于黄线左侧ready_out_buf为0,输出端阻塞，因此关闭输入端，保持在输出态



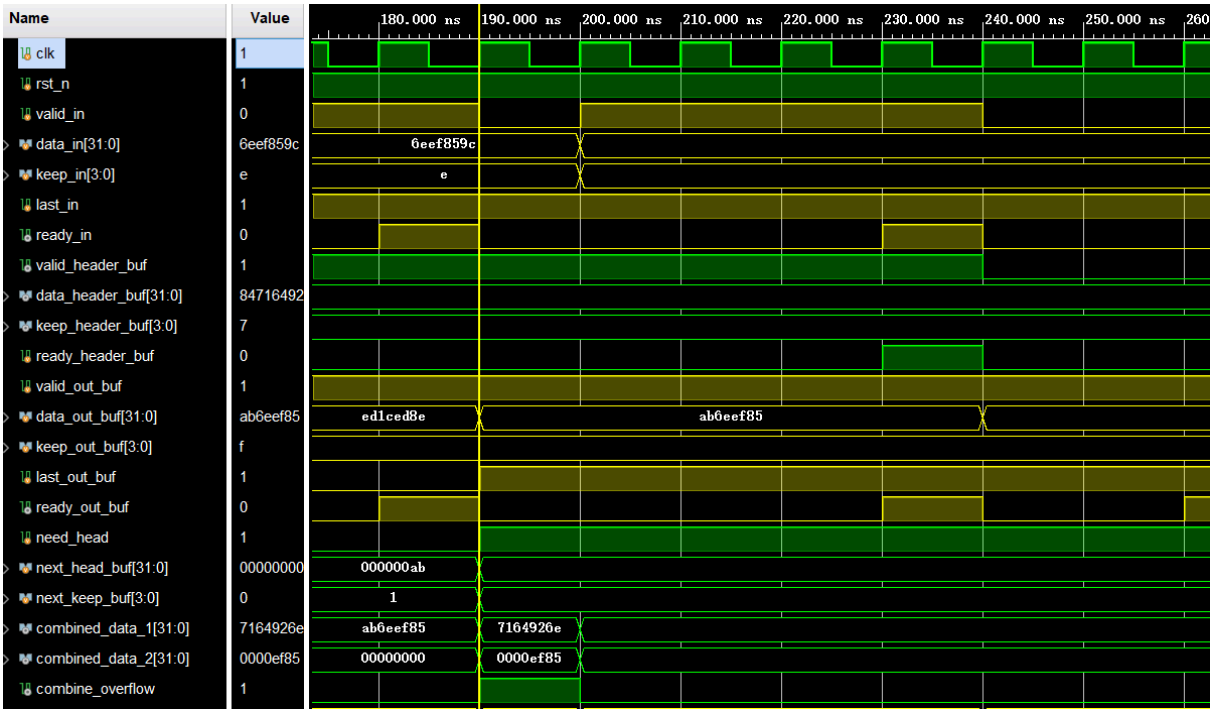
3.2.8输出态→输出态（情况2）

黄线左侧ready_out_buf为1，该帧数据成功输出，同时valid_in为1新的一帧到来，且不是最后一帧，因此下一周期仍为输出态，data_out_buf为0xed1ced8e正确



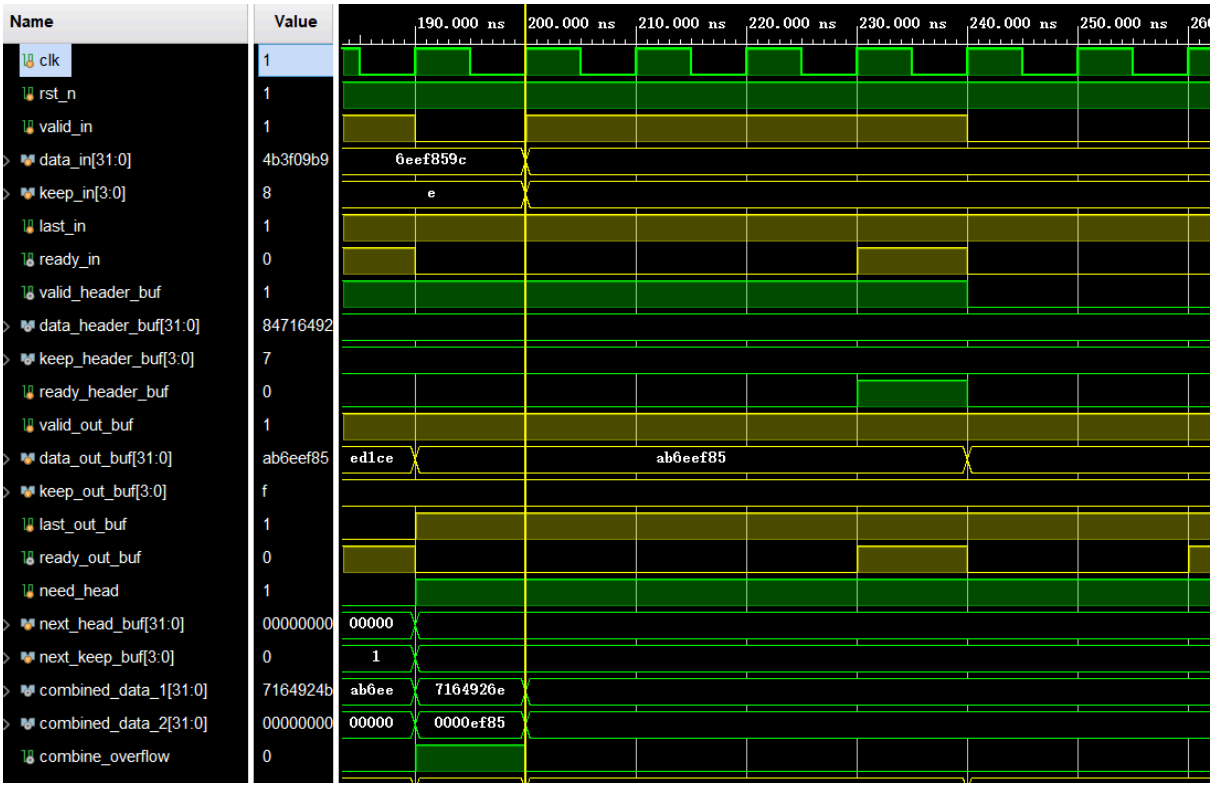
3.2.9输出态→结尾输出态2

黄线左侧ready_out_buf为1，该帧数据成功输出，同时valid_in为1新的一帧到来且为最后一帧，并且上一帧未输出部分(0xab)和该帧(0x6eef85)可一次性输出，进入结尾输出态2



3.2.10结尾输出态2→结尾输出态2

黄线左侧ready_out_buf为0，输出端阻塞，因此关闭输入端，保持在结尾输出态2



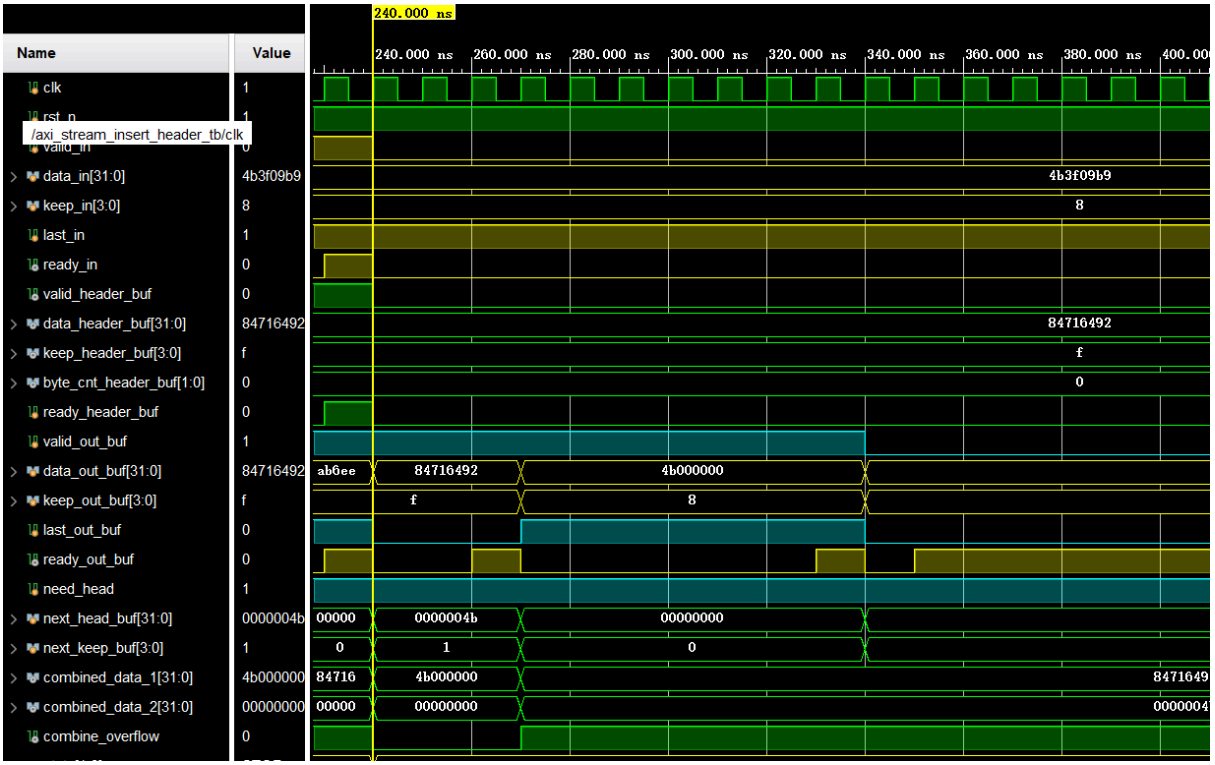
3.2.11结尾输出态2→结尾输出态2（情况2）

黄线左侧ready_out_buf为1，该帧数据成功输出，也标志着该数据流输出完成；此时新的header和data已经到来，且data为最后一帧，data和header可以一次性输出，进入结尾输出态2，data_out_buf为0x7164924b正确

以上过程通过随机数据测试获得，为了验证剩余情况，采用构造数据

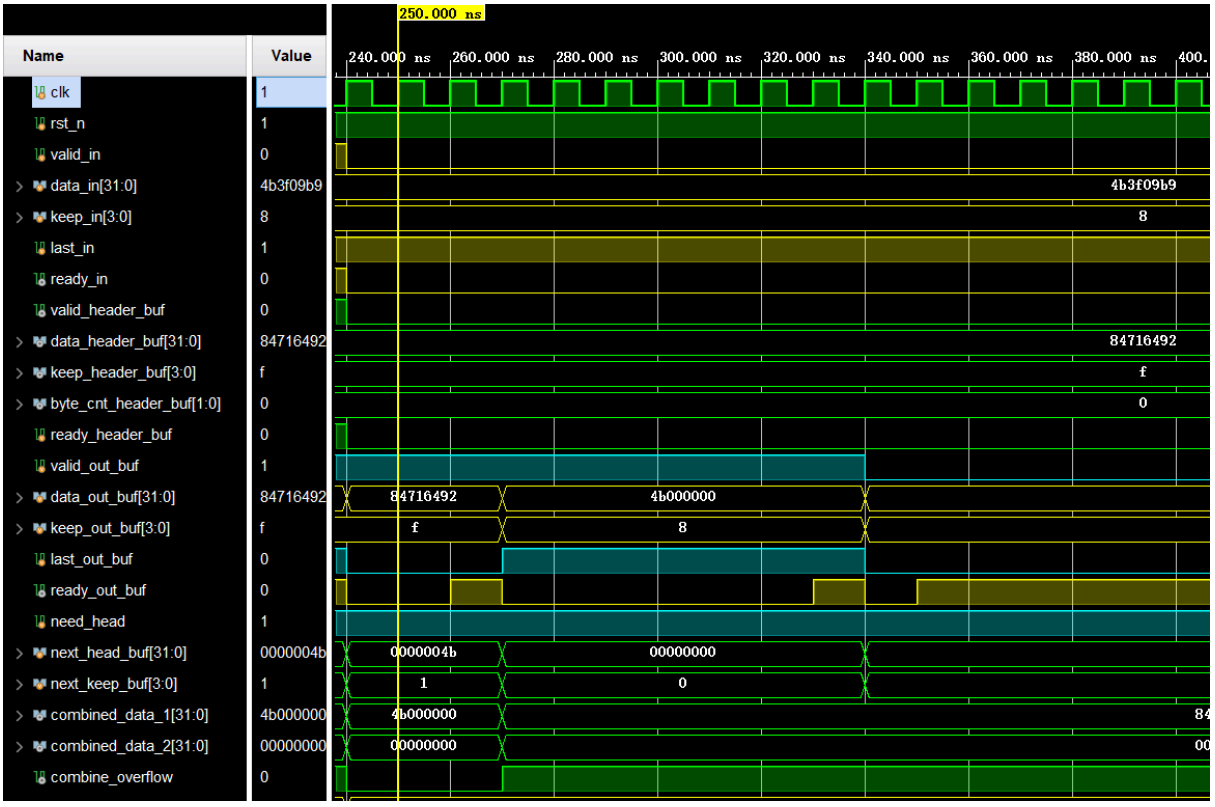
3.2.12结尾输出态2→结尾输出态1

黄线左侧ready_out_buf为1，该帧数据成功输出，也标志着该数据流输出完成；此时新的header和data已经到来，且data为最后一帧，data和header不可以一次性输出，进入结尾输出态1，data_out_buf为0x84716492正确



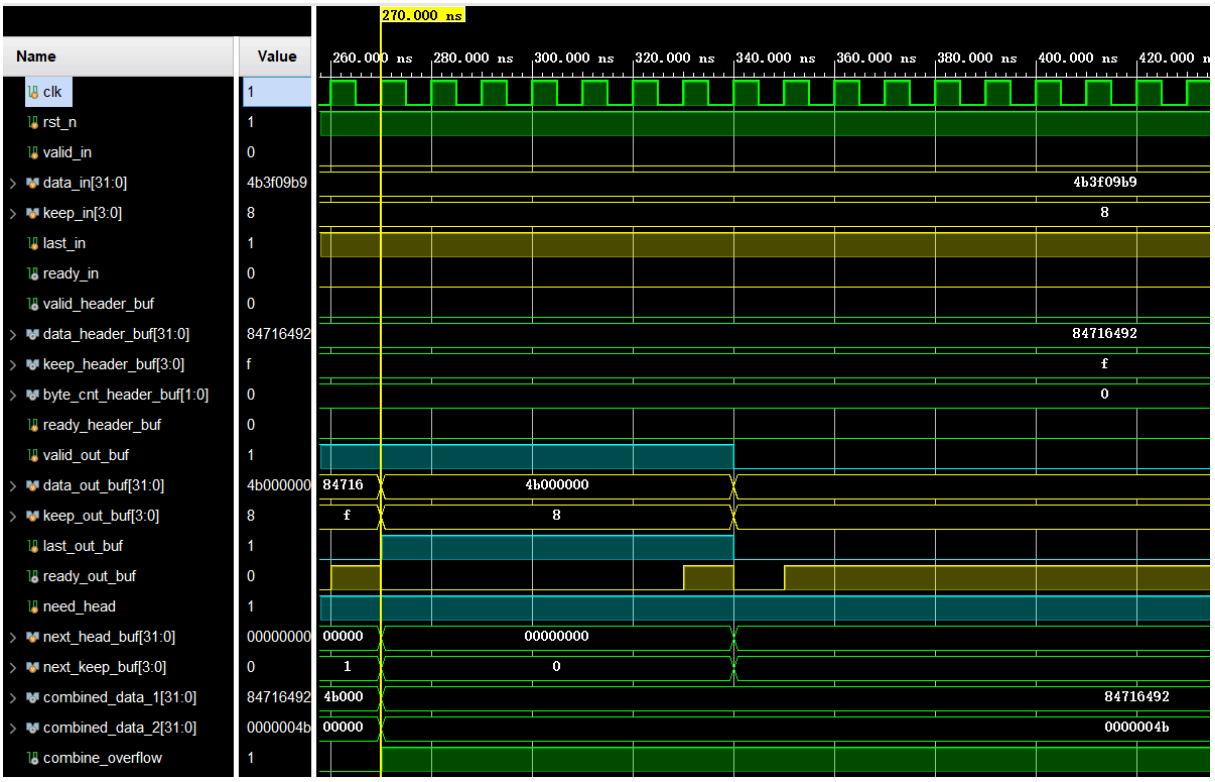
3.2.13结尾输出态1→结尾输出态1

黄线左侧ready_out_buf为0，输出端阻塞，因此关闭输入端，保持在结尾输出态1



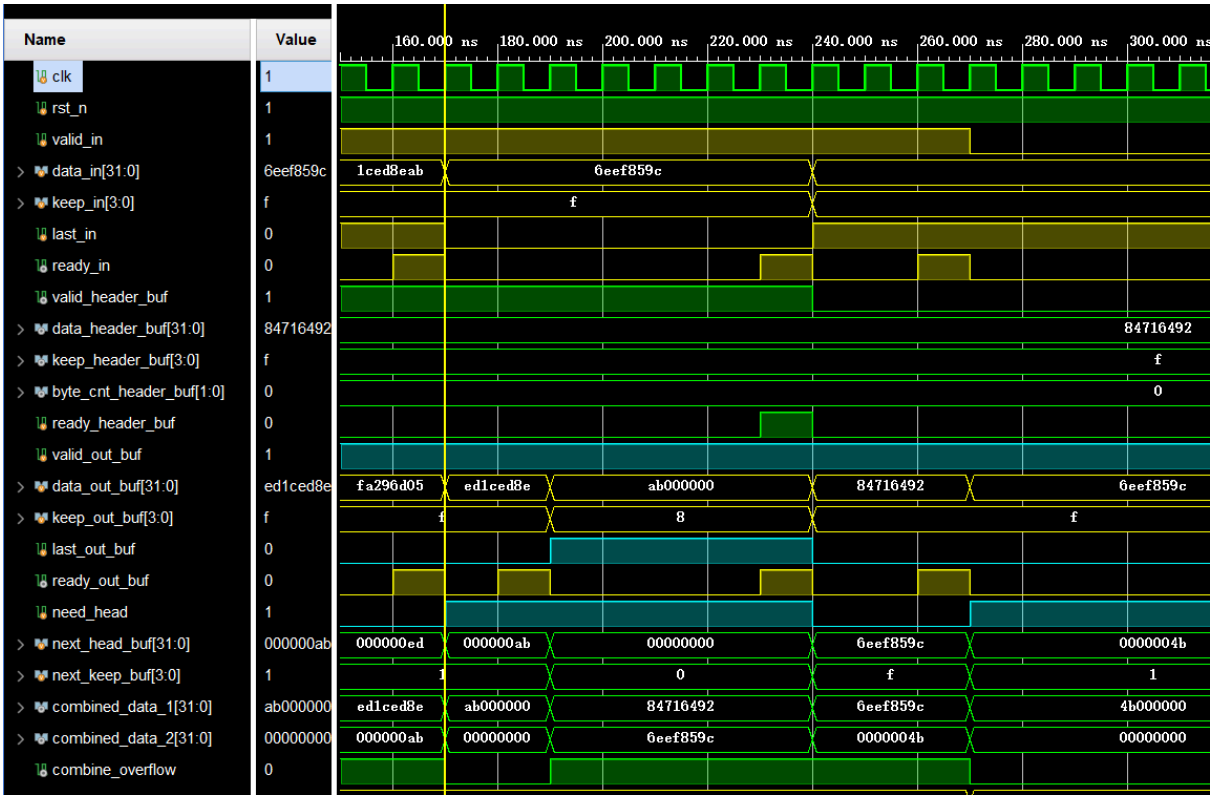
3.2.14结尾输出态1→结尾输出态2

黄线左侧ready_out_buf为1，该帧数据成功输出，剩余结尾部分输出，进入结尾输出态2，data_out_buf为0x4b正确



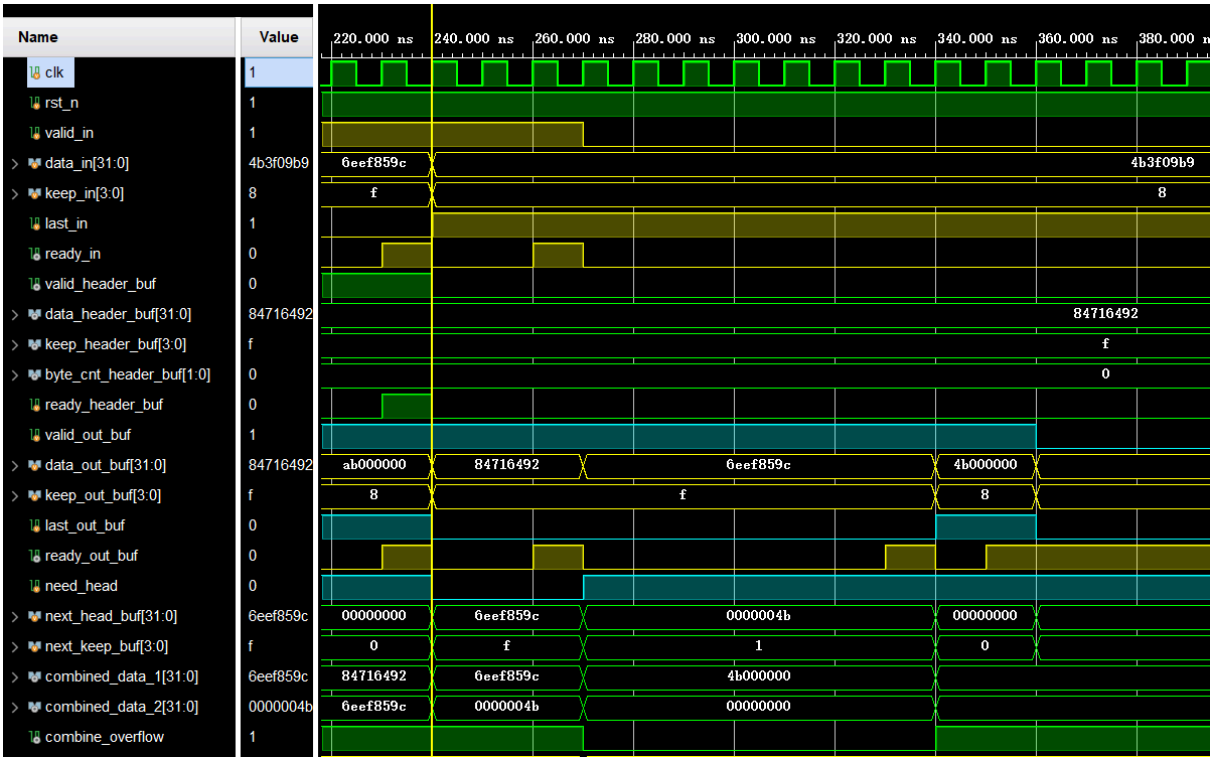
3.2.15输出态→结尾输出态1

黄线左侧ready_out_buf为1，该帧数据成功输出，同时valid_in为1新的一帧到来且为最后一帧，并且上一帧未输出部分(0xed)和该帧(0x1ced8eab)不可一次性输出，进入结尾输出态1



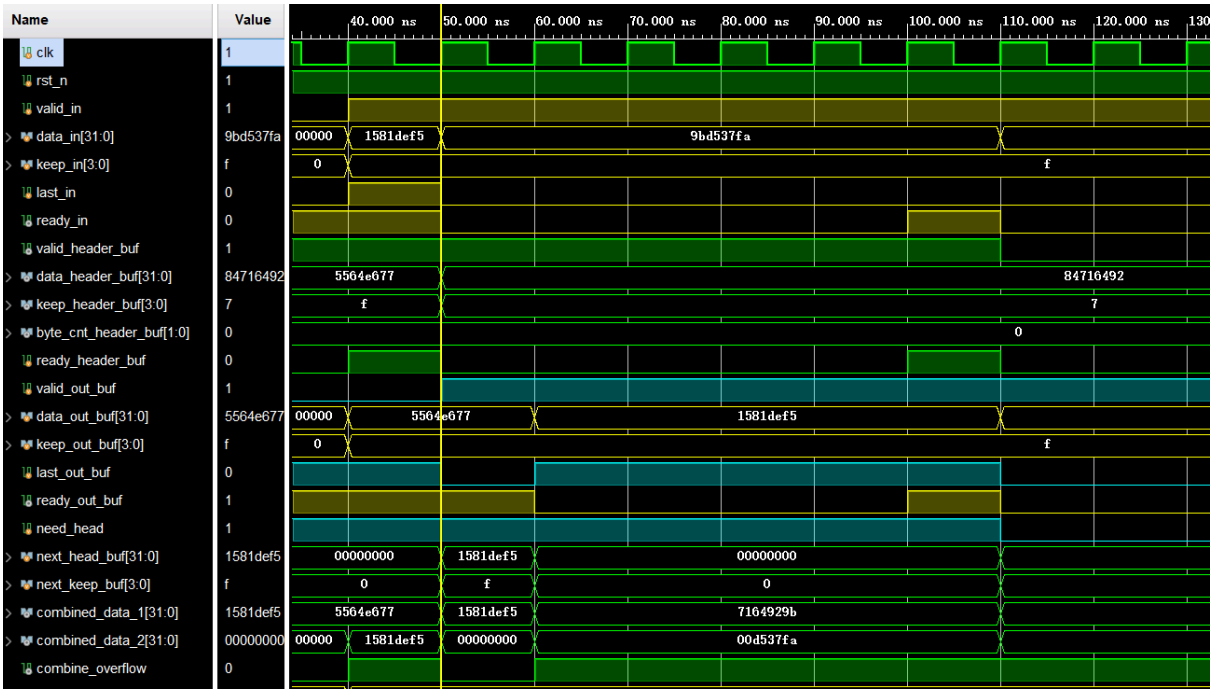
3.2.16结尾输出态2→输出态

黄线左侧ready_out_buf为1，该帧数据成功输出，也标志着该数据流输出完成；此时新的header和data已经到来，且data不是最后一帧，进入输出态，data_out_buf为0x84716492正确



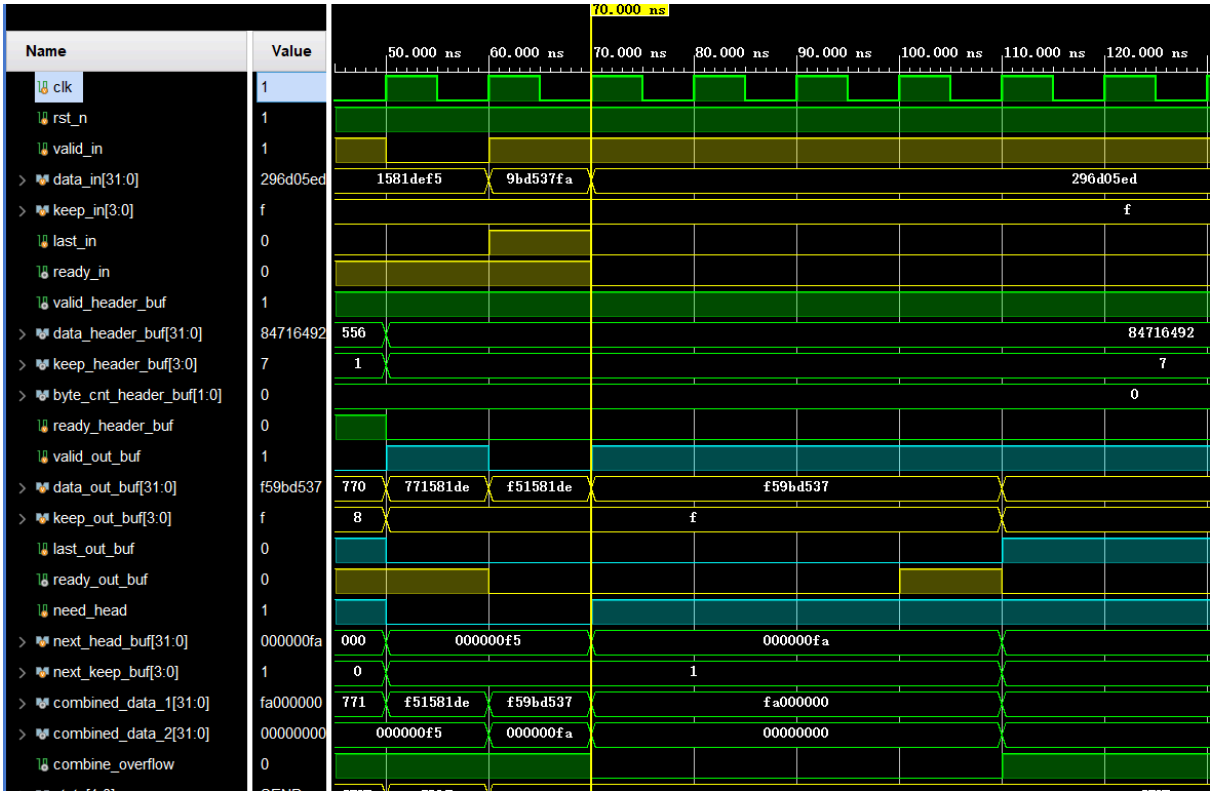
3.2.17 idle态→结尾输出态1

黄线左侧周期，header和data均已到达，data已经是最后一帧，header0x5564e677和data0x1581def5无法一次性输出，进入结尾输出态1，data_out_buf为0x5564e677正确



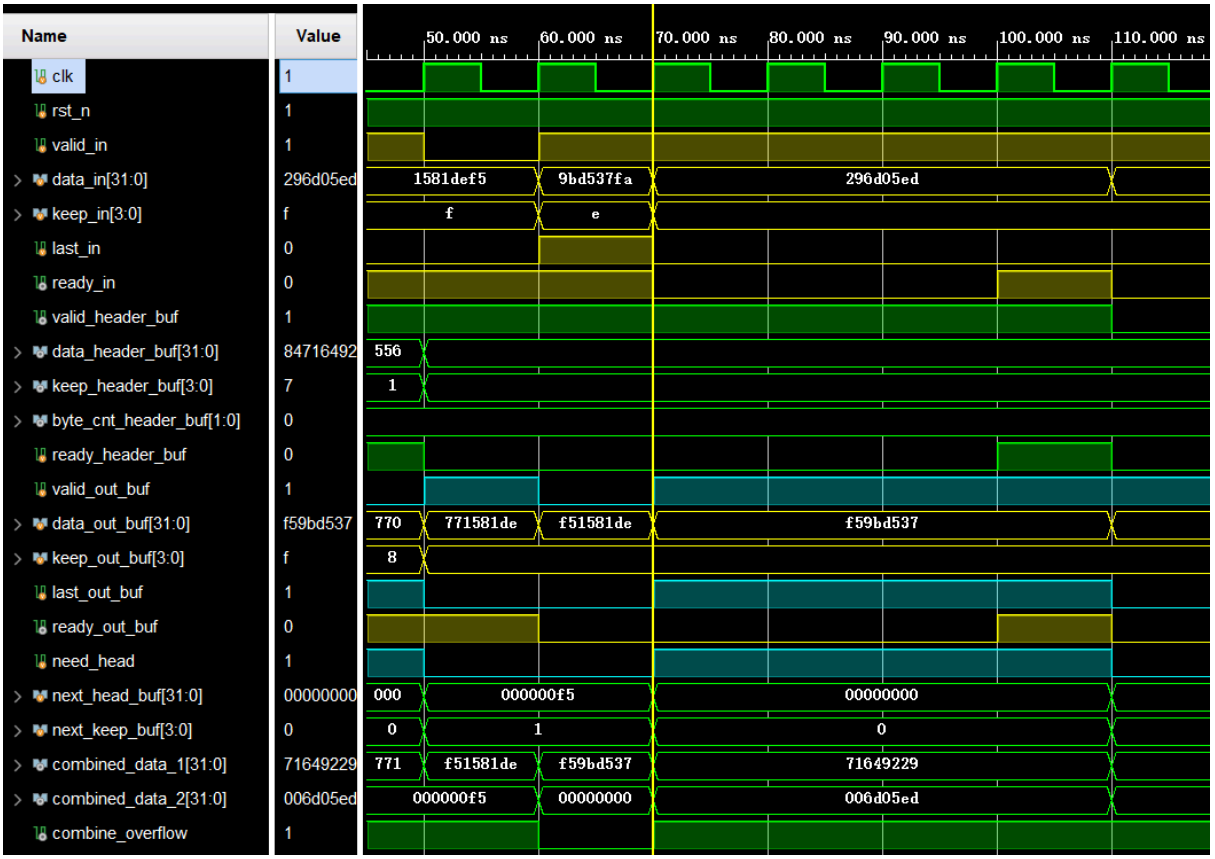
3.2.18 暂停态→结尾输出态1

黄线左侧周期，数据输入恢复，且是最后一帧数据，剩余未输出部分0xf5和最后一帧0x9bd537fa无法一次性发送，进入结尾输出态1，data_out_buf为0xf59bd537正确



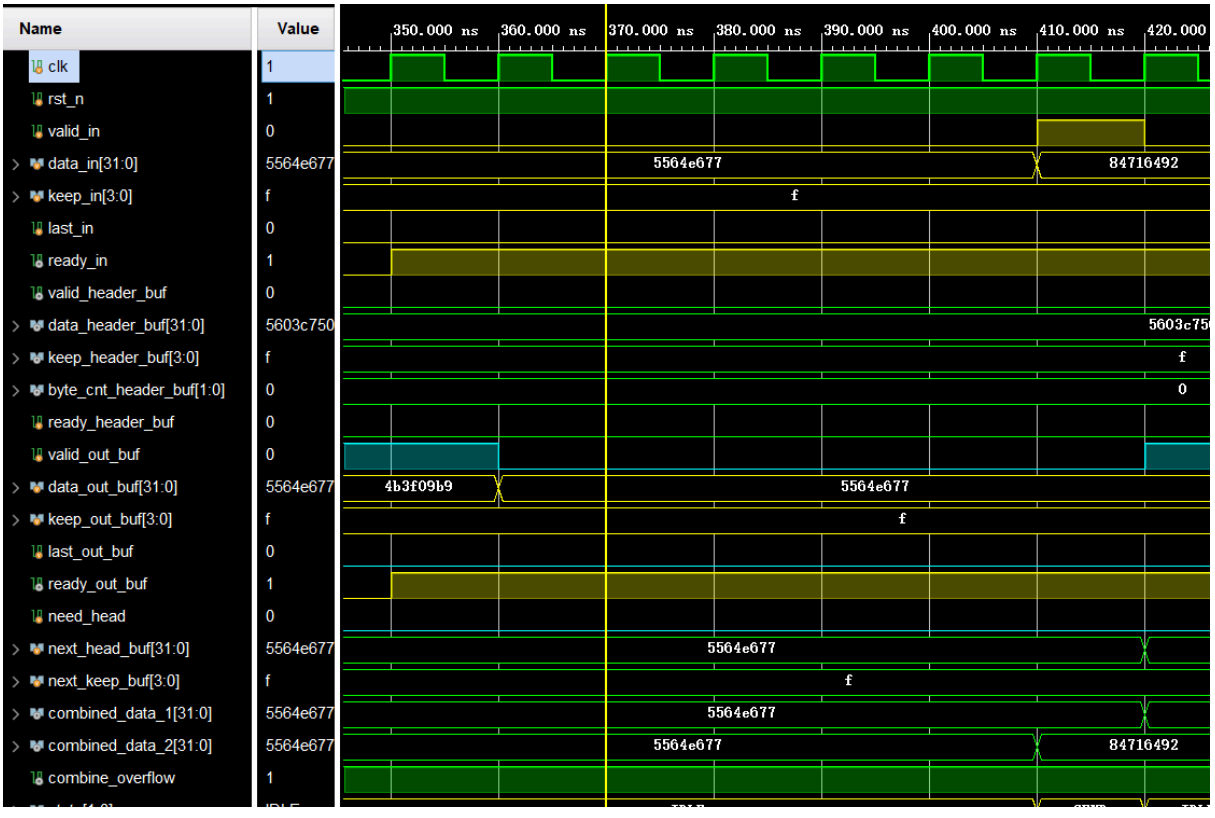
3.2.19 暂停态→结尾输出态2

黄线左侧周期，数据输入恢复，且是最后一帧数据，剩余未输出部分0xf5和最后一帧0x9bd537可以一次性发送，进入结尾输出态2，data_out_buf为0xf59bd537正确



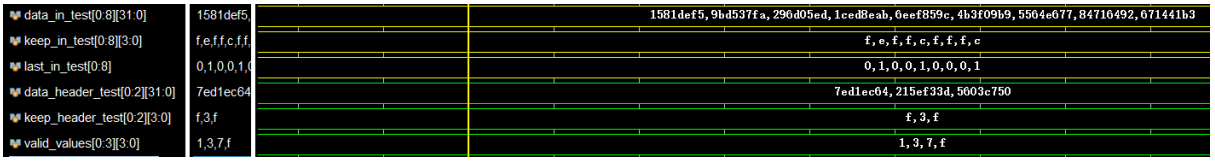
3.2.20 暂停态→暂停态

黄线左侧周期，valid_in为0，数据输入仍未恢复，保持暂停态



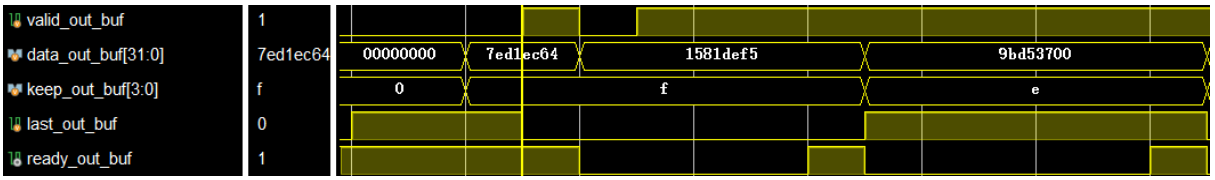
3.3 最终效果波形图（以DATA_WD=32,HEADER_NUM=3为例）

输入data和header序列如下

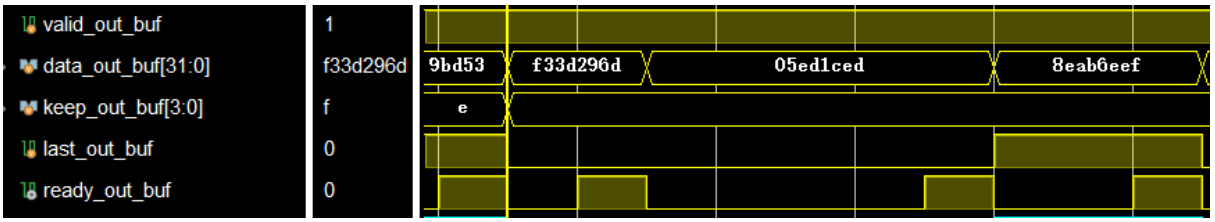


输出data序列如下，与输入相符

第一次输出



第二次输出



第三次输出