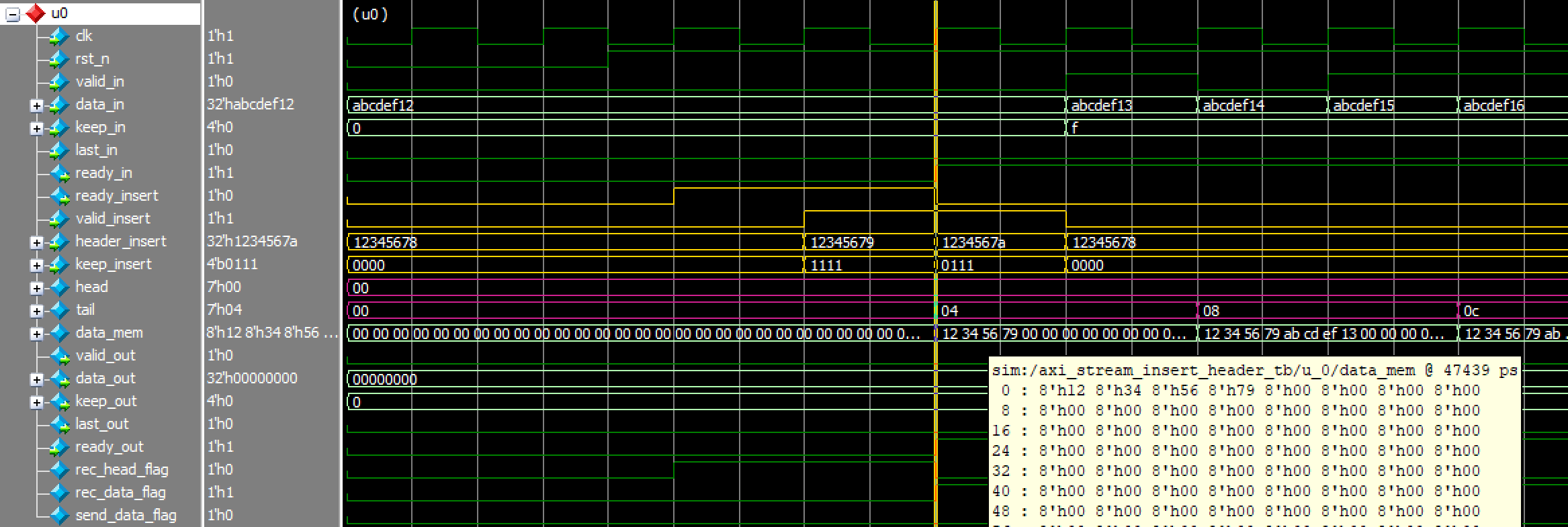
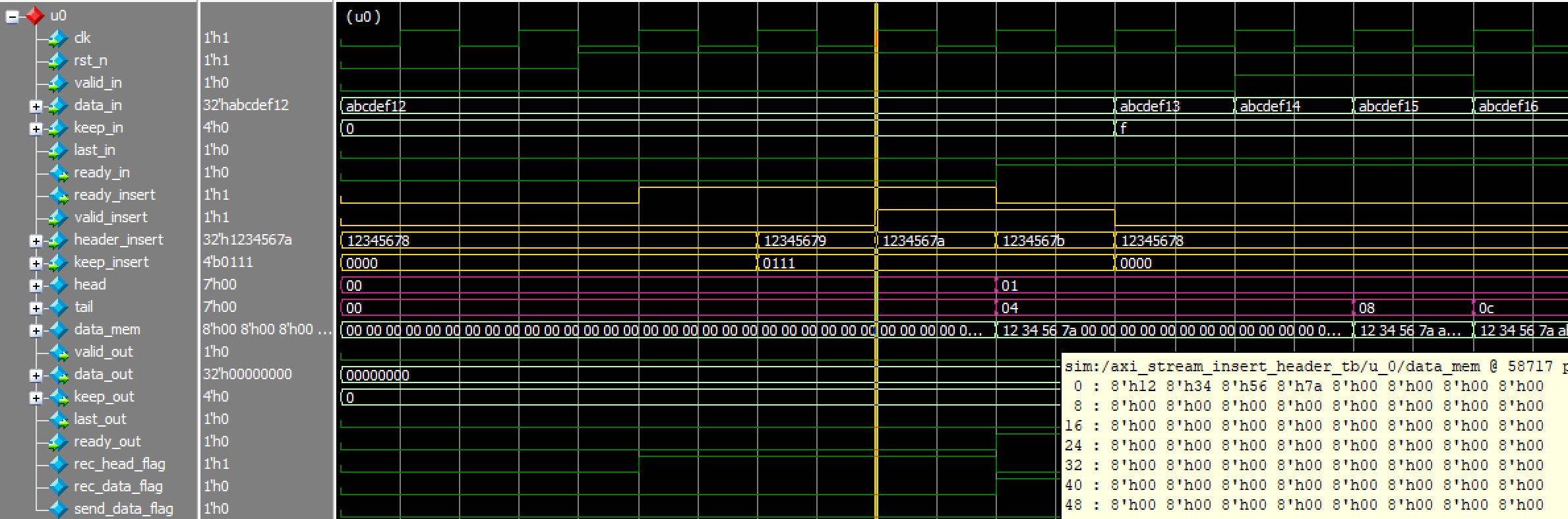
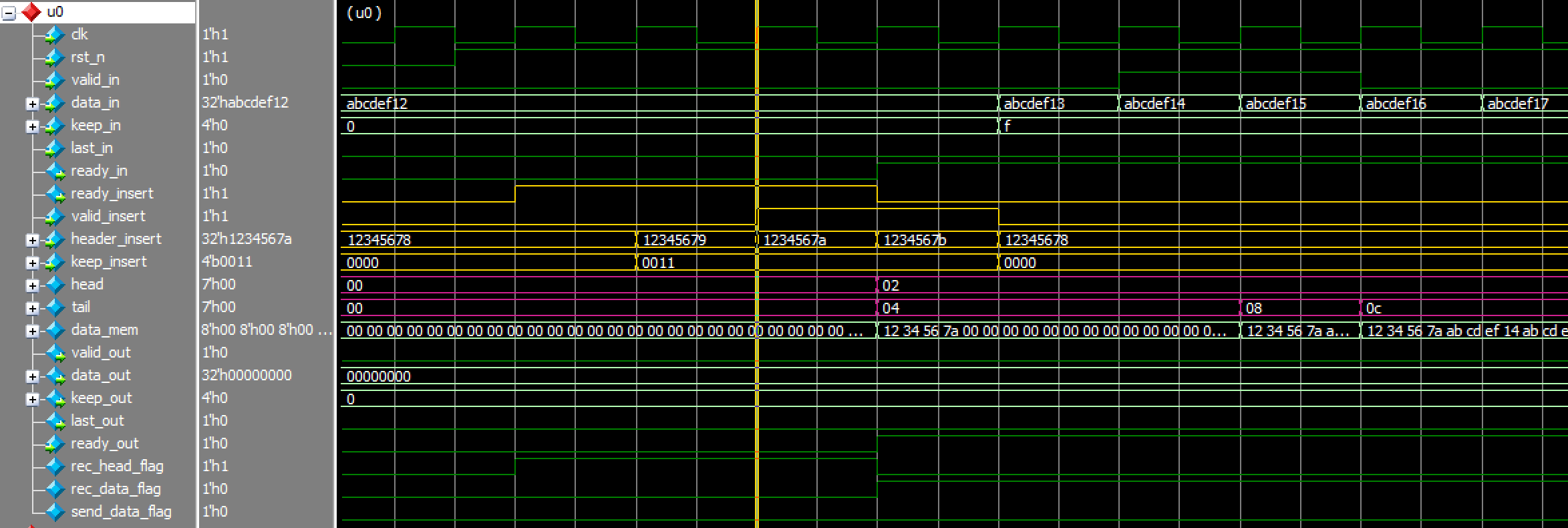
**仿真结果**

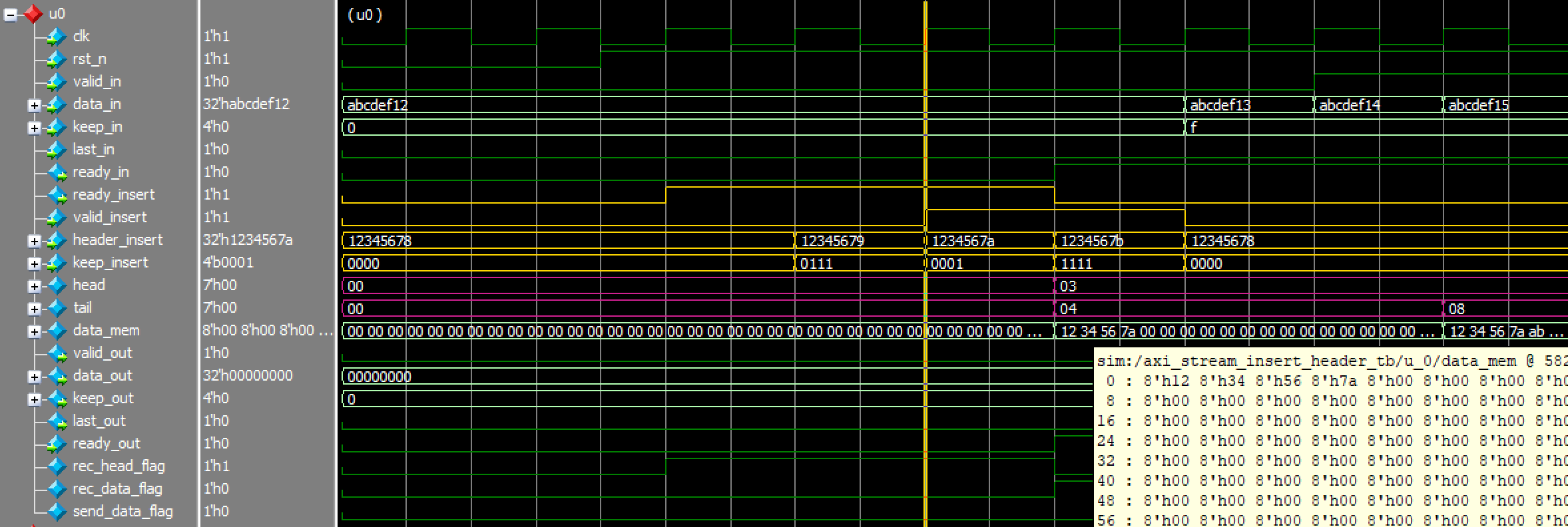
# 仿真说明

## 工况1



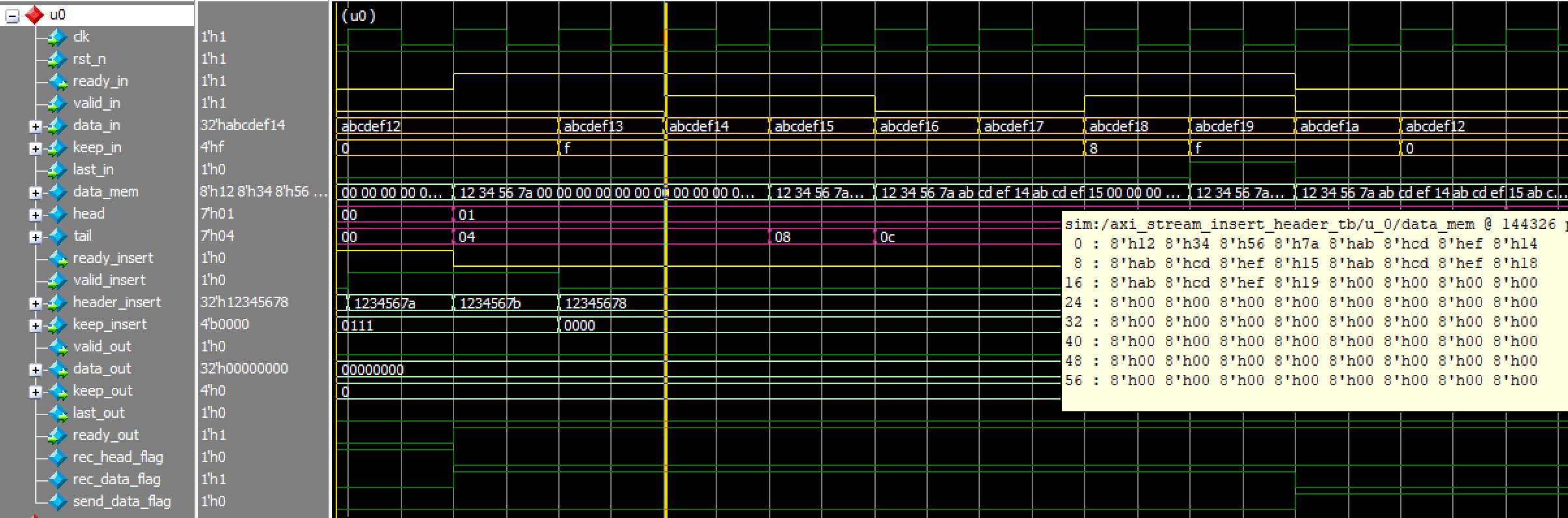






如图所示，当ready\_insert和valid\_insert都为‘1’时，将header\_insert存入data\_mem中，并且根据keep\_insert更新head和tail，keep\_insert为“1111”时，head为0，发送数据是从地址0开始发；keep\_insert为“0111”时，head为1，发送数据是从地址1开始发；keep\_insert为“0011”时，head为2，发送数据是从地址2开始发；keep\_insert为“0001”时，head为3，发送数据是从地址3开始发。

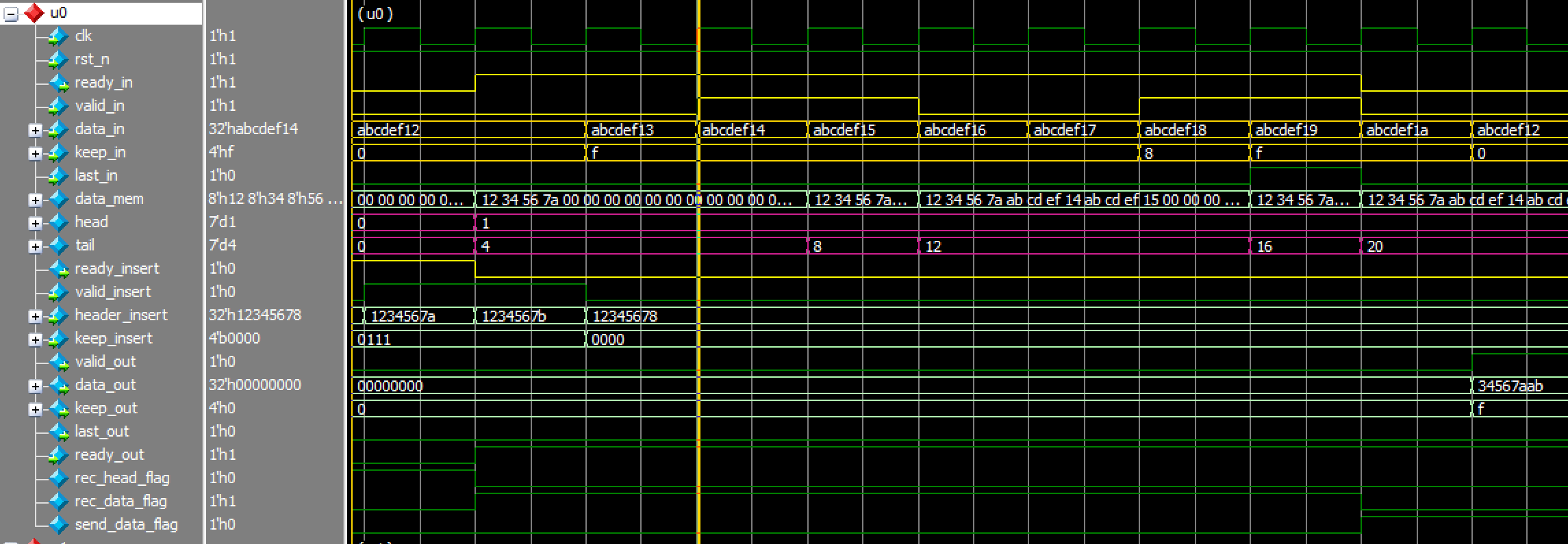
## 工况2



3

2

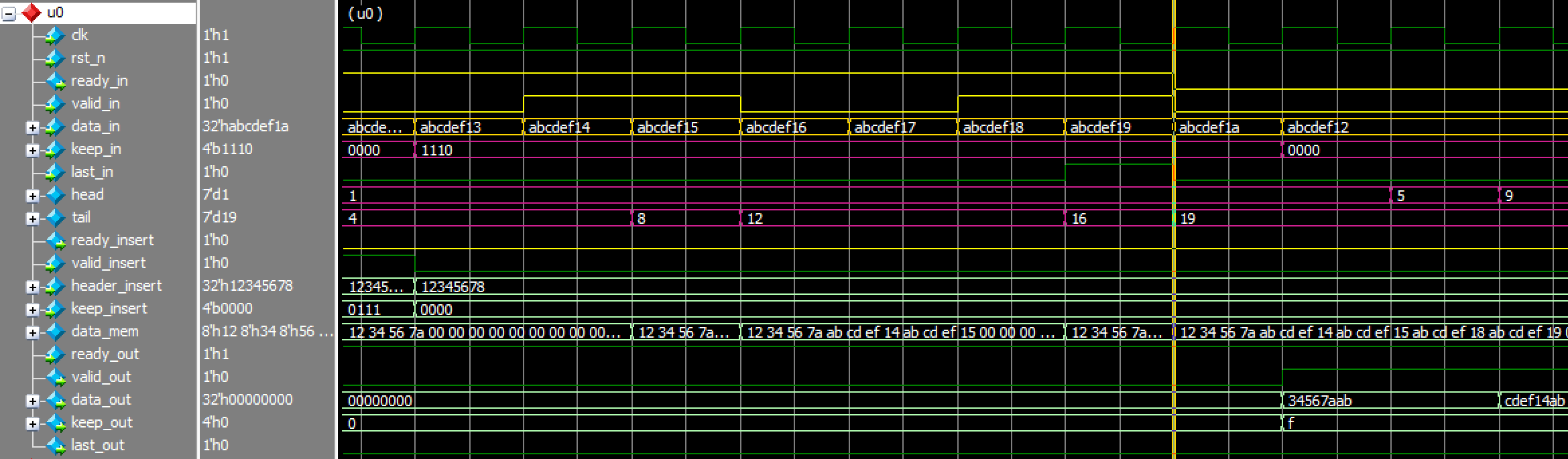
1

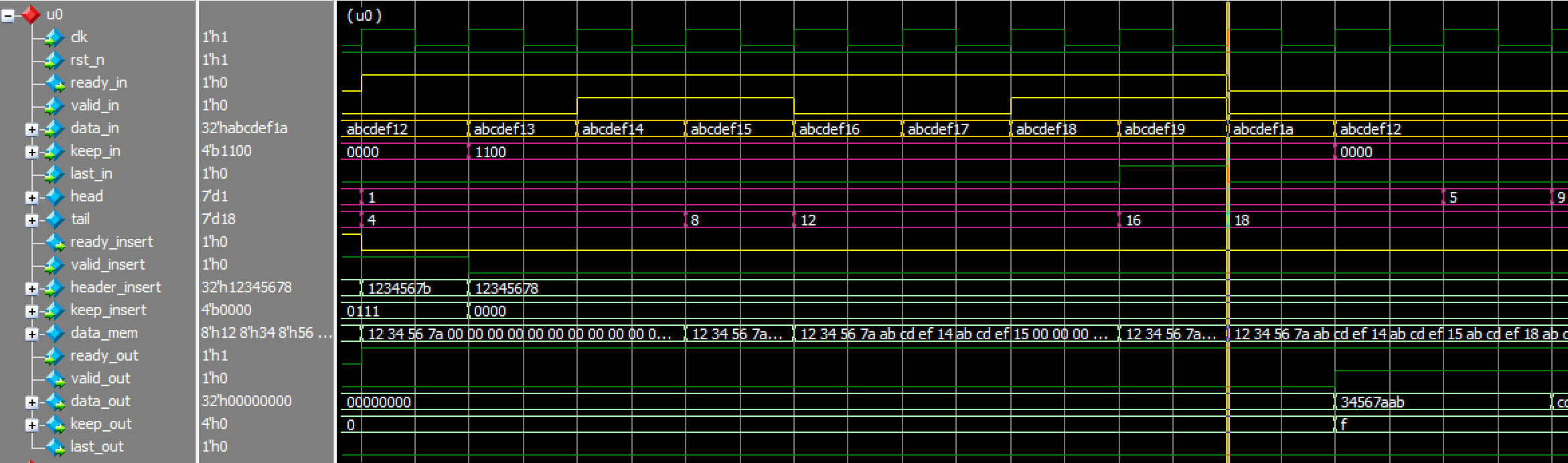


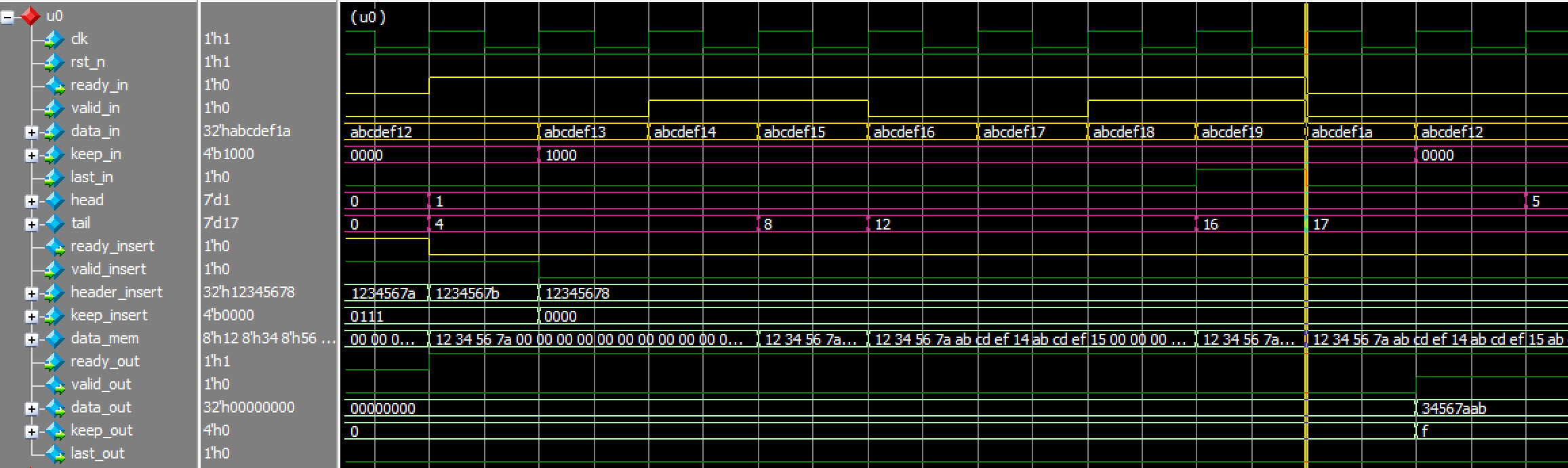
5

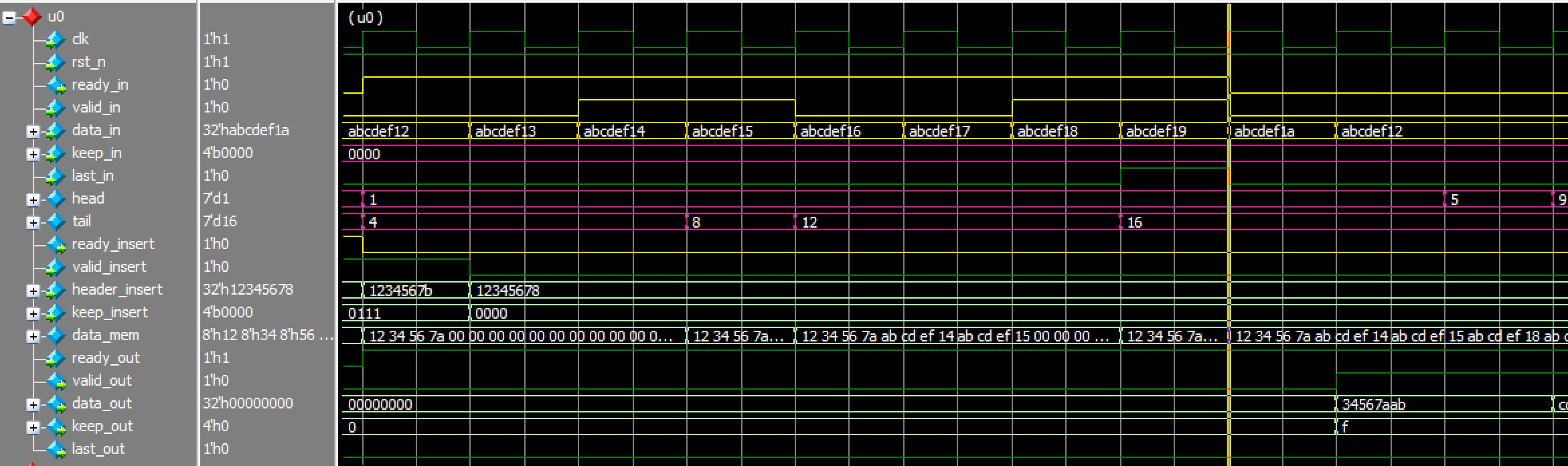
4

如图所示，由Mark1可以看出，ready\_insert拉低时，ready\_in拉高，开始接收数据部分；由Mark2、Mark3、Mark4可以看出，当ready\_in和valid\_in都为高时，将对应的data\_in存入data\_mem，并且修改计算寄存器长度的tail地址信号，一共存了4个数据，加上帧头，共5个数据，所以tail为20；通过Mark3可以看出，data\_mem里存储的数据与ready\_in和valid\_in都为高时的data\_in数据是一一对应的；由Mark5可以看出，当last\_in拉高时，代表该数据为本次传输最后一个数据，将ready\_in拉低，停止接收数据。



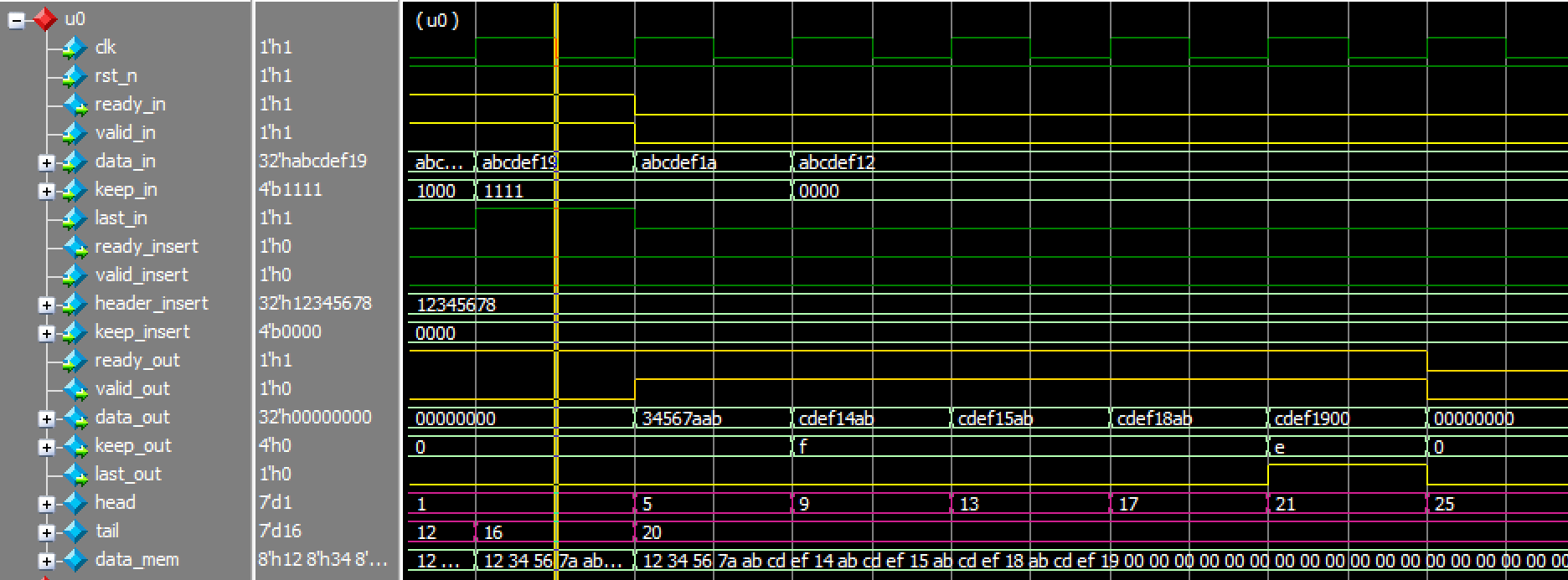






如图所示，当last\_in拉高时，需要通过keep\_in判断有效位，当keep\_in为1110时，最后一字节无效，tail为19；当keep\_in为1100时，最后两字节无效，tail为18；当keep\_in为1000时，最后三字节无效，tail为17；当keep\_in为0000时，所有字节无效，tail为16。

## 工况3



3

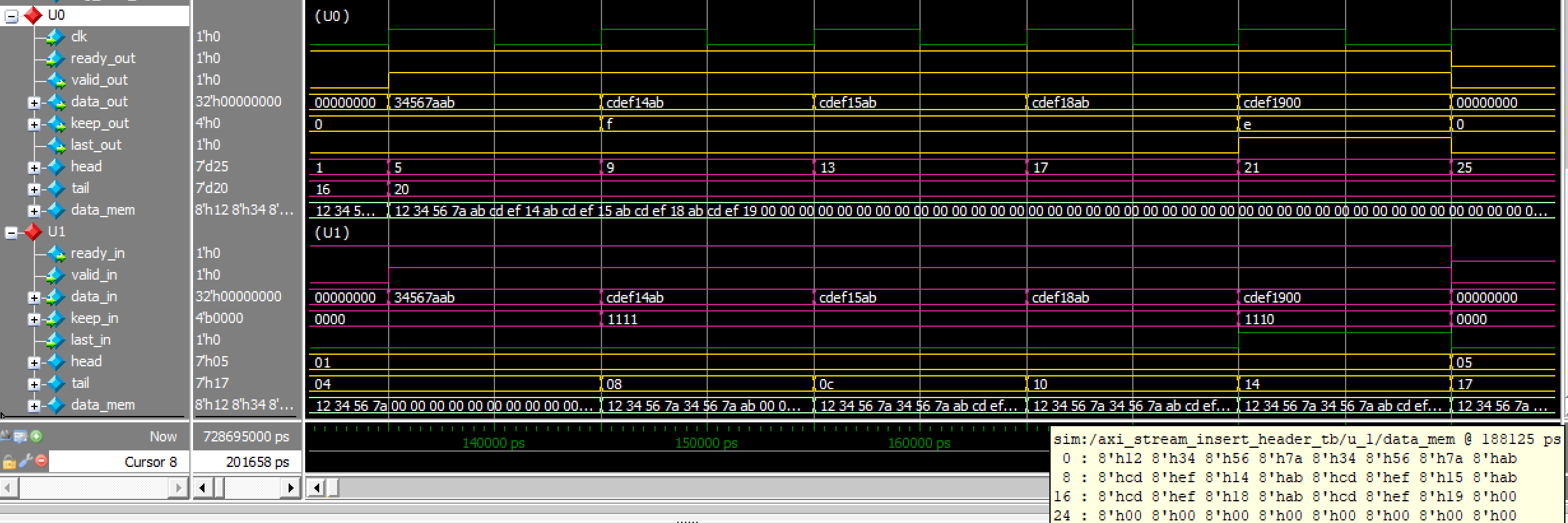
2

1

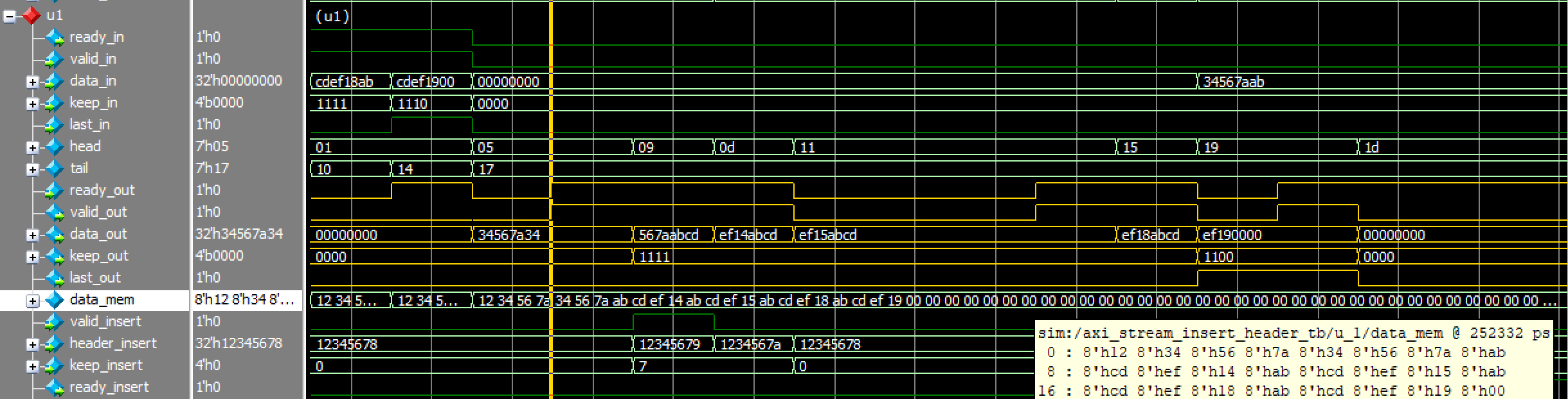
e=1110

如图所示，由Mark1可以看出，valid\_in拉低之后，valid\_out开始拉高，发送给数据，由于提前将数据放在寄存器中，可以直接输出，没有气泡产生；由Mark2可以看出，由于第一字节数据无效，从第二字节开始输出，data\_out与data\_mem中的数据能够一一对应；由Mark3可以看出，由于最后一个数据最后一字节无效，keep\_out为1110，并且valid\_out发送完最后一个数据后拉低。

## 工况4



如图所示，将U0与U1级联，U0的输出作为U1的数据输入，可以得到如上图所示的data\_mem，下面主要关注U1的输出。



如图所示，valid\_out为高时的数据与data\_mem中的数据一致，并且在ready\_out拉低后再拉高时，不会丢失数据，也不会重复传输数据。