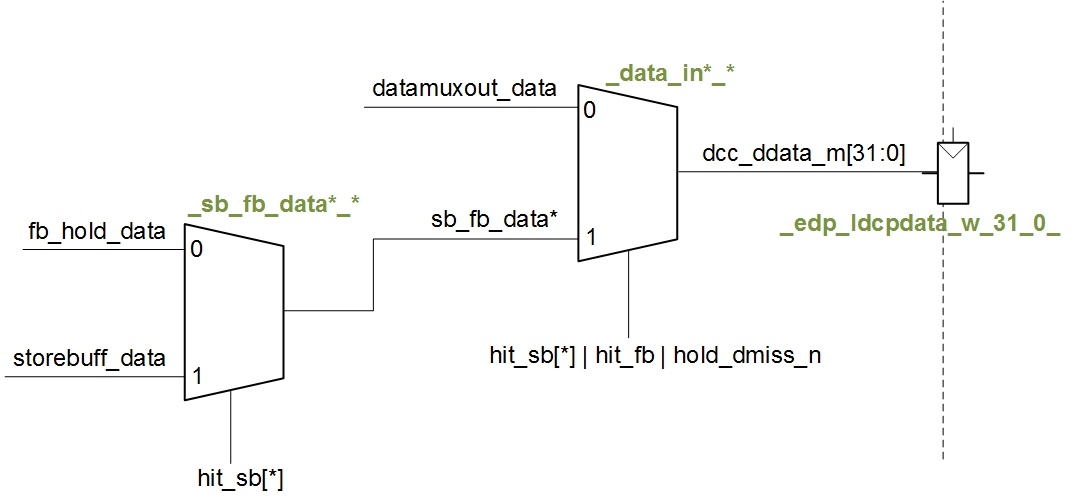
**Sequence 1.**

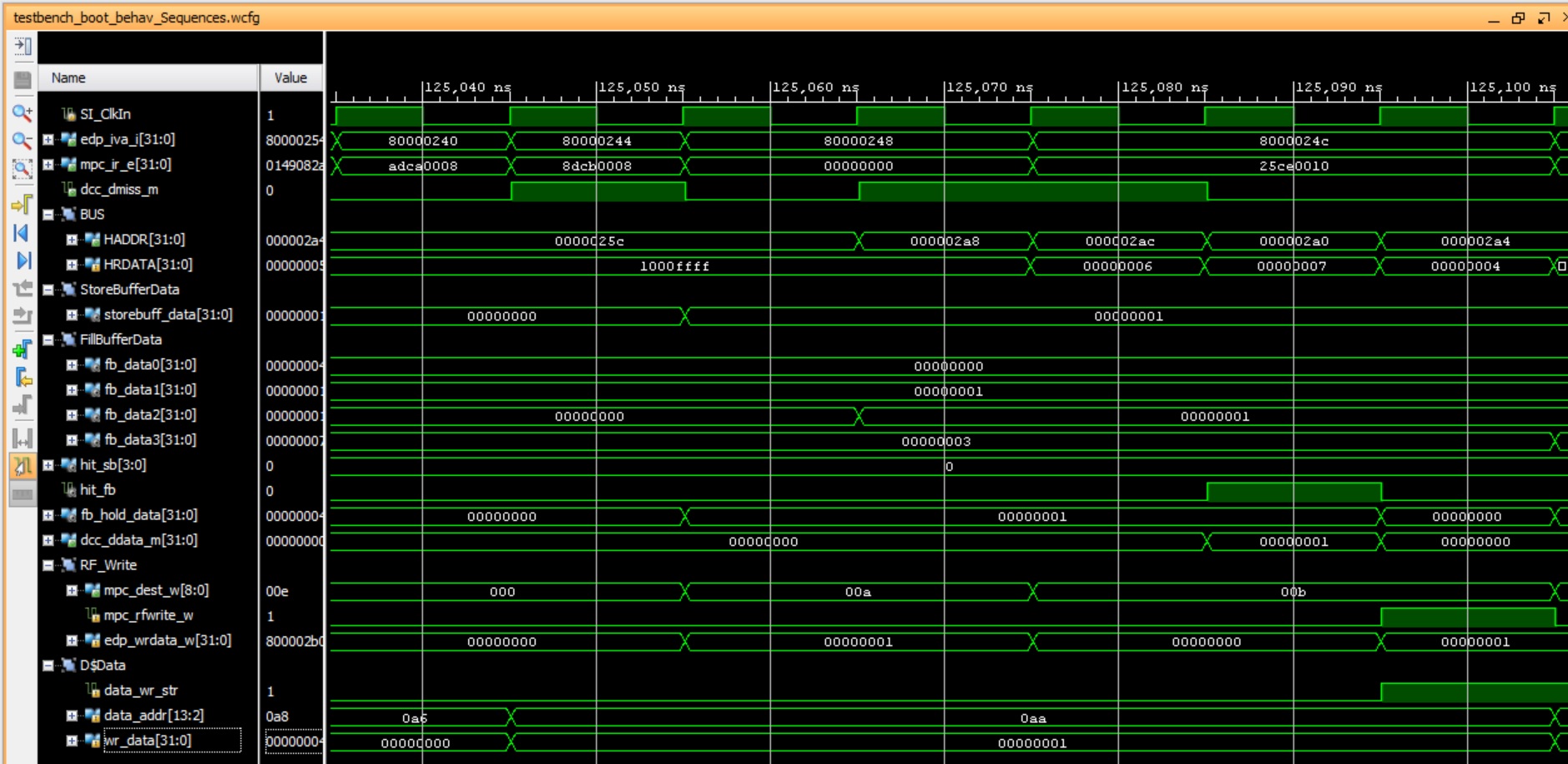
In lines 2197-2205 of module m14k\_dcc, the following multiplexers are implemented:



The value read by the lw (*dcc\_ddata\_m[31:0]*) can come from several places, such as the D$ (*datamuxout\_data[31:0]*), the Store Buffer (*storebuff\_data[31:0]*) or the Fill Buffer (*fb\_hold\_data[31:0]*). The results of the simulation, for the second iteration of the loop, are shown in the figure at the next page. In that figure:

* 1st cycle: The sw instruction is at the E-Stage (mpc\_ir\_e=0xadca0008).
* 2nd cycle:
  + The sw instruction is at the M-Stage. A miss is detected.
  + The lw instruction is at the E-Stage (mpc\_ir\_e=0x8dcb0008).
* 3rd cycle: The word to write by the store is written in the Store Buffer (*storebuff\_data*=0x1).
* 4th cycle: The word to write by the store is transferred from the Store Buffer into the Fill Buffer (*fb\_data2*=0x1).
* 6th cycle: A hit is detected in the Fill Buffer for the word requested by the lw instruction (hit\_fb=1). Thus, in the multiplexers illustrated above, the data read by the lw is provided from the Fill Buffer: *dcc\_ddata\_m*=*fb\_hold\_data[31:0]*=0x1.
* 7th cycle:
  + The inputs to write the value read by the lw into the register file are computed (*mpc\_dest\_w*=0x00b, *mpc\_rfwrite\_w*=1, *edp\_wrdata\_w*=0x1).
  + Also, in this cycle, the D$ is started to be written with the new block.

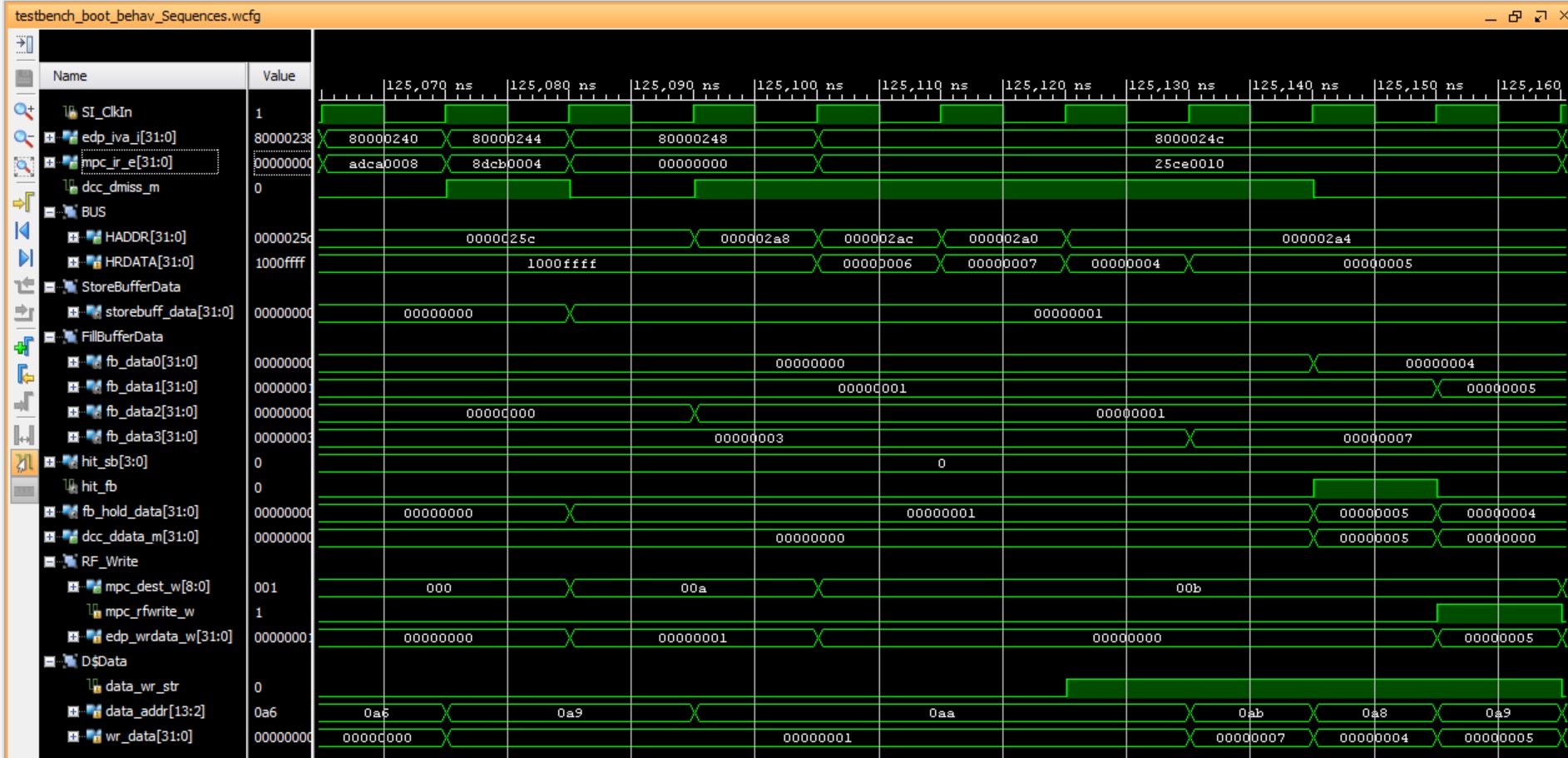
Note that the miss penalty for the lw is reduced or even removed completely.



**Sequence 2.**

In this case, the word is provided to the load from the Fill Buffer, as in the previous example. However, the lw has to wait for the requested word to arrive through the bus. The results of the simulation are shown in the next figure.

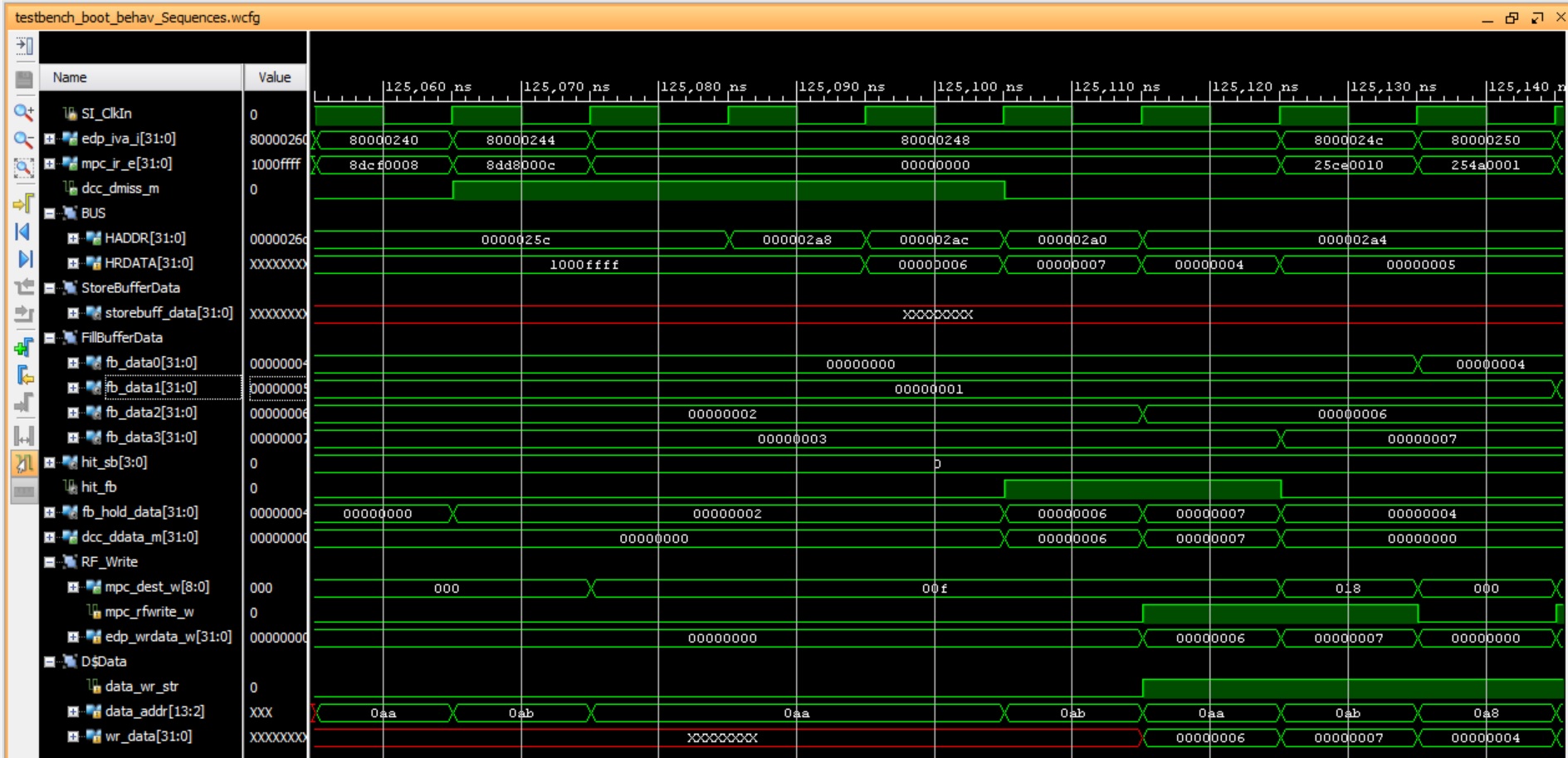
Given that the word written by the store is the third word in the line, and the word requested by the load is the second word in the line, which is the last word to arrive through the bus, there is a miss penalty for the load instruction.



**Sequence 3.**

The first lw pays the miss penalty (4 cycles), but the second lw receives the requested word directly from the Fill Buffer, thus it pays a smaller or even non-existent penalty. The results of the simulation are provided in the next figure.

Note that the miss penalty for the second lw is reduced or even removed completely.



**Sequence 4.**

A cacheable load that misses blocks until its critical word is returned from the BIU. When the word returns, the load completes, and the next instruction enters the M-stage. In this sequence, the next instruction is a store, which accesses the same physical address line as the load, but a different word. It first stalls, because it initially misses and does not have access to the bus, then hits as its critical word (a non-critical word from the first load) returns from the BIU. The word written by the store is written to the Store Buffer and eventually updated in the D$.

