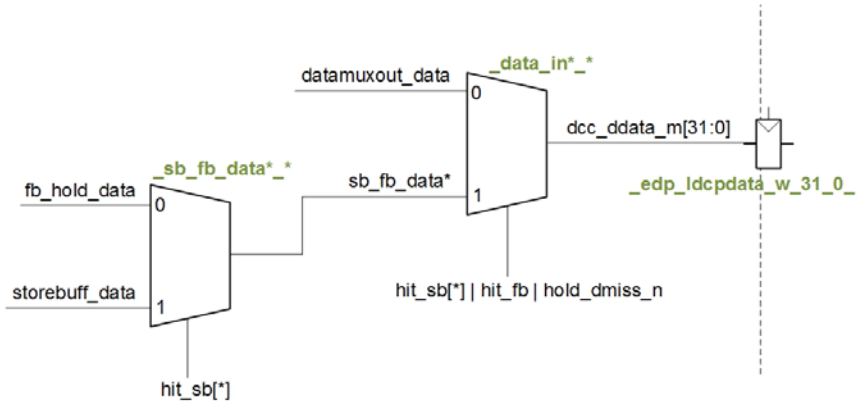
Sequence 1.

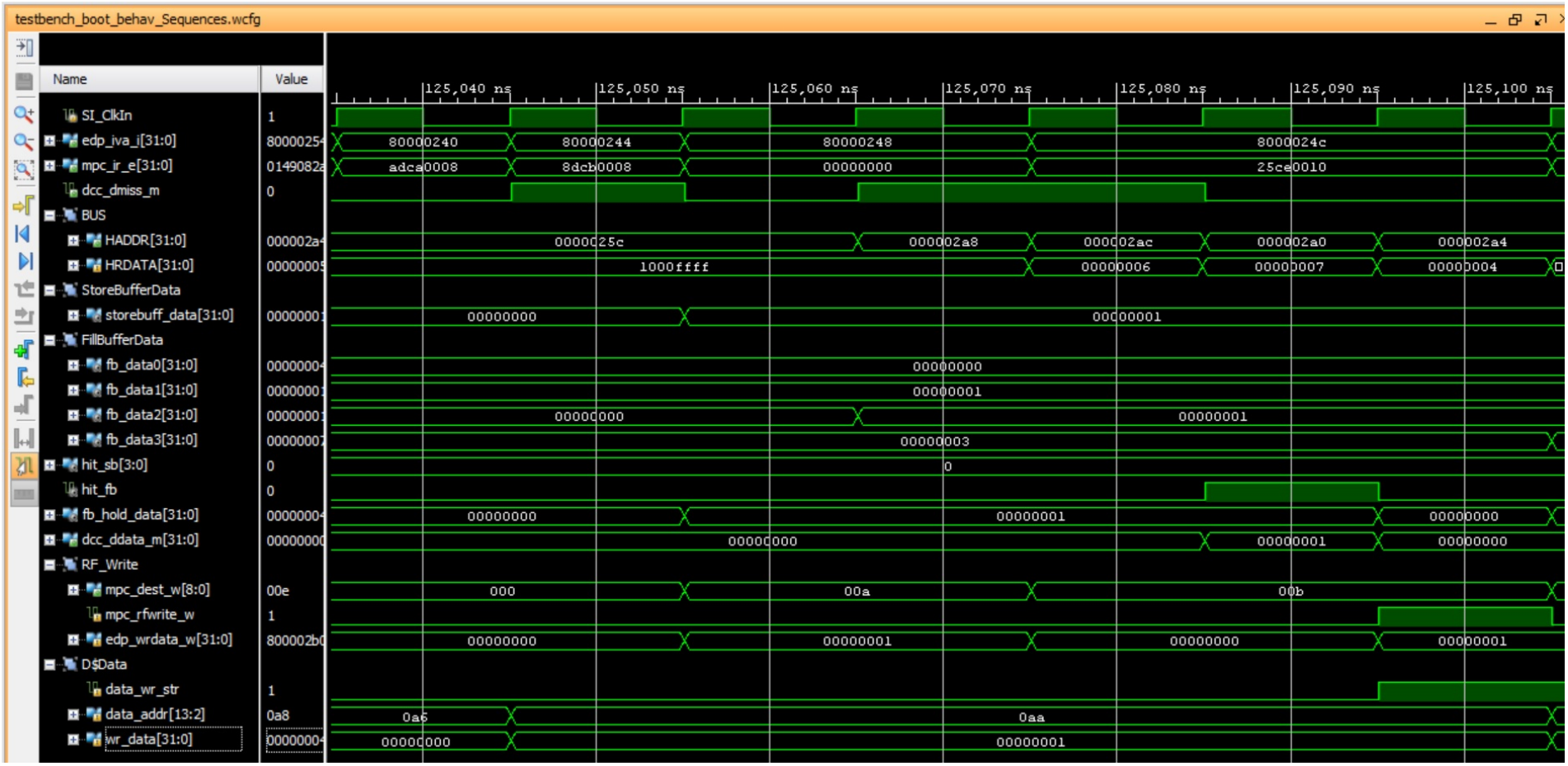
In lines 2197-2205 of module m14k_dcc, the following multiplexers are implemented:



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.

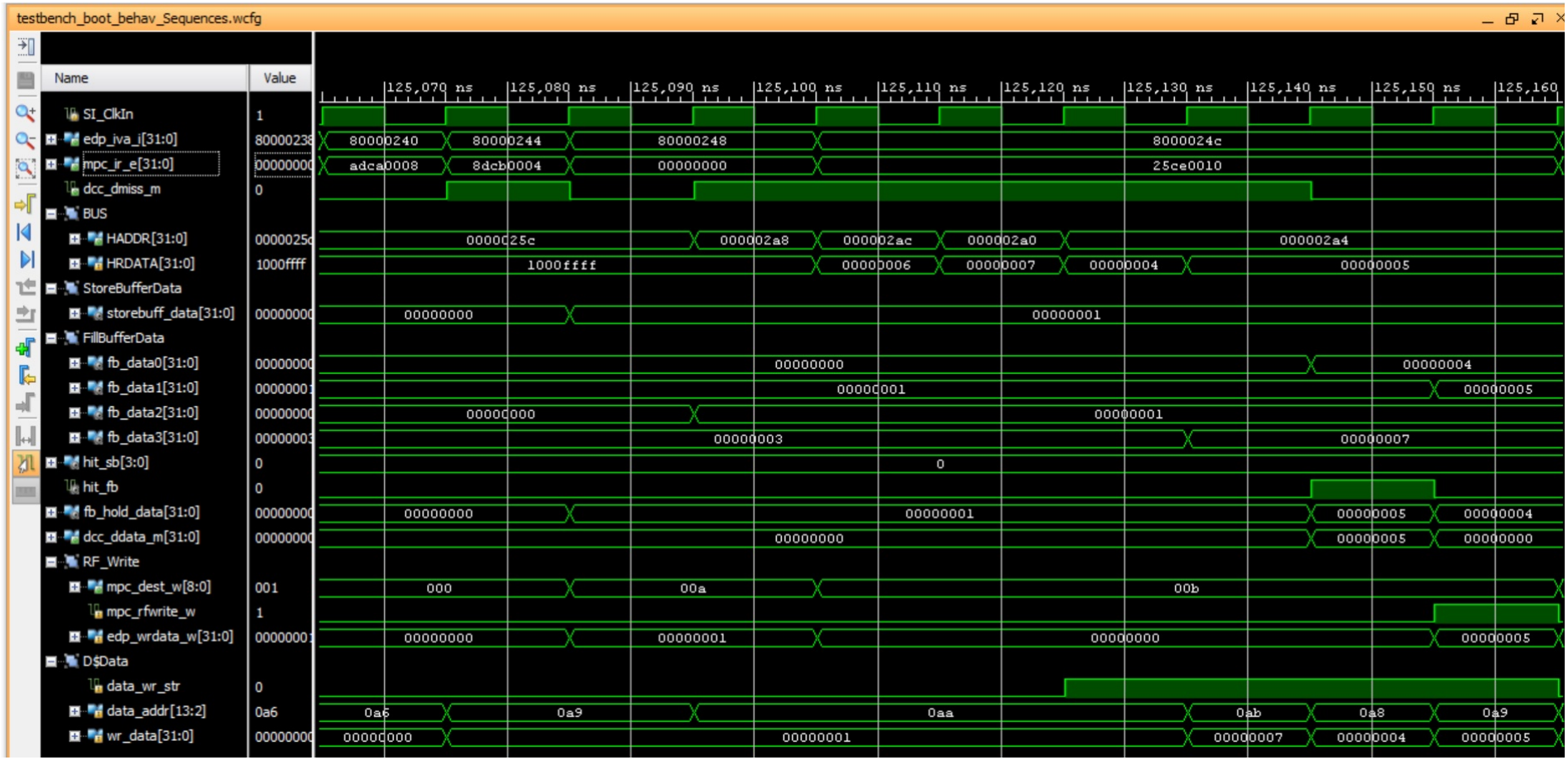
Note that the miss penalty for the lw is reduced or even removed completely.


testbench_boot_behav_Sequences.wcfg
Name
Value
125,040 ns
125,050 ns
125,060 ns
125,070 ns
125,080 ns
125,090 ns
125,100 ns
SI_ClkIn
edp_iva_i[31:0]
mpc_ir_e[31:0]
dcc_dmiss_m
BUS
HADDR[31:0]
HRDATA[31:0]
StoreBufferData
storebuff_data[31:0]
FillBufferData
fb_data0[31:0]
fb_data1[31:0]
fb_data2[31:0]
fb_data3[31:0]
hit_sb[3:0]
hit_fb
fb_hold_data[31:0]
dcc_ddata_m[31:0]
RF_Write
mpc_dest_w[8:0]
mpc_rfwrite_w
edp_wrdata_w[31:0]
D$Data
data_wr_str
data_addr[13:2]
wr_data[31:0]

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.

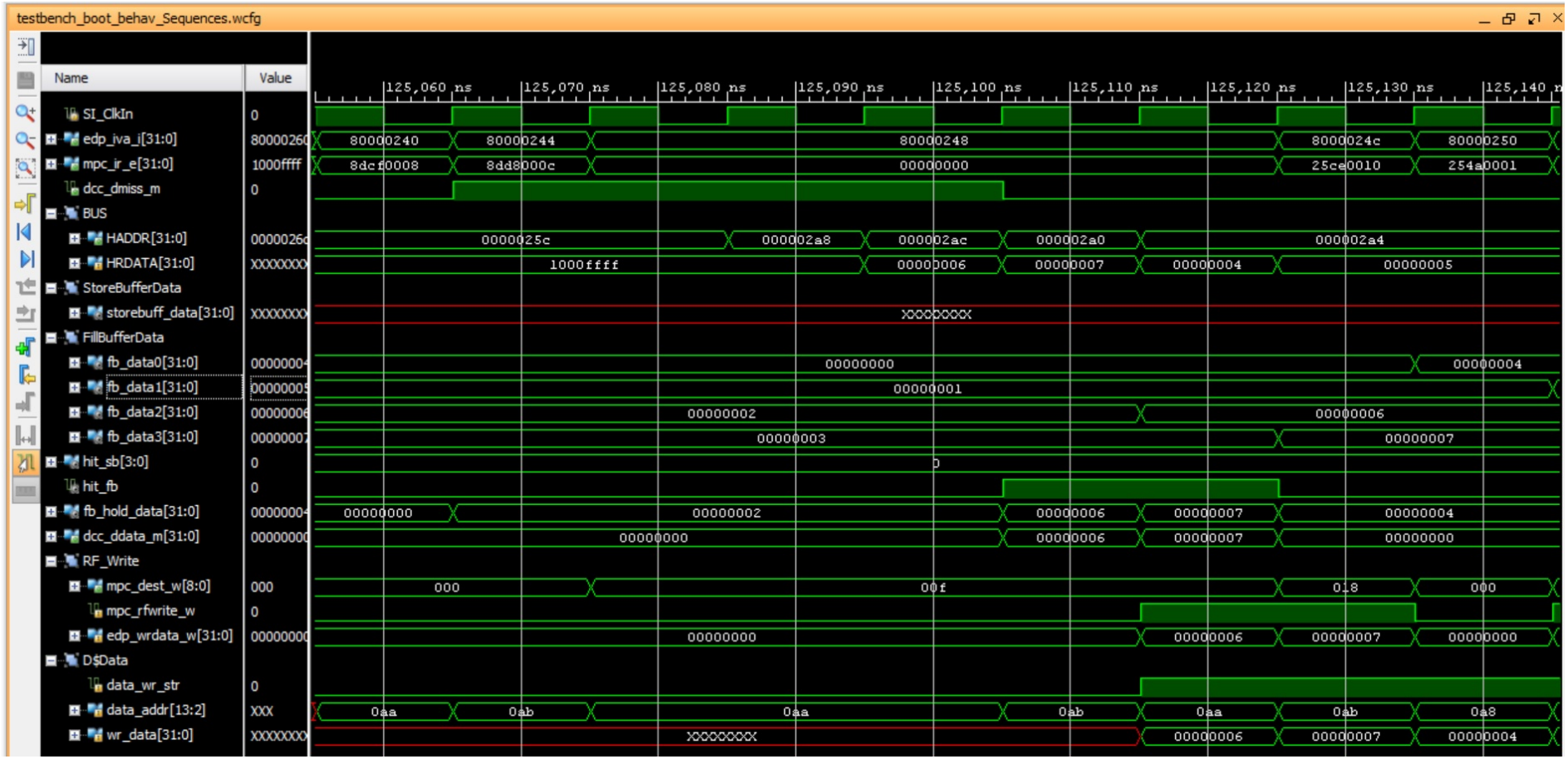
Given that the word written by the store is the third word in the line, and the word requested by the load is the second word in the line, which is the last word to arrive through the bus, there is a miss penalty for the load instruction.


testbench_boot_behav_Sequences.wcfg
Name
Value
125,070 ns
125,080 ns
125,090 ns
125,100 ns
125,110 ns
125,120 ns
125,130 ns
125,140 ns
125,150 ns
125,160
SI_ClkIn
1
edp_iva_i[31:0]
80000240
80000244
80000248
8000024c
mpc_ir_e[31:0]
adca0008
8dcb0004
00000000
25ce0010
dcc_dmiss_m
0
BUS
HADDR[31:0]
0000025c
000002a8
000002ac
000002a0
000002a4
HRDATA[31:0]
1000ffff
00000006
00000007
00000004
00000005
StoreBufferData
storebuff_data[31:0]
00000000
00000001
FillBufferData
fb_data0[31:0]
00000004
fb_data1[31:0]
fb_data2[31:0]
fb_data3[31:0]
00000003
00000007
hit_sb[3:0]
hit_fb
fb_hold_data[31:0]
dcc_ddata_m[31:0]
RF_Write
mpc_dest_w[8:0]
001
000
00a
00b
mpc_rfwrite_w
edp_wrdata_w[31:0]
D$Data
data_wr_str
data_addr[13:2]
0a6
0a9
0aa
0ab
0a8
0a9
wr_data[31:0]
00000007
00000004

Sequence 3.

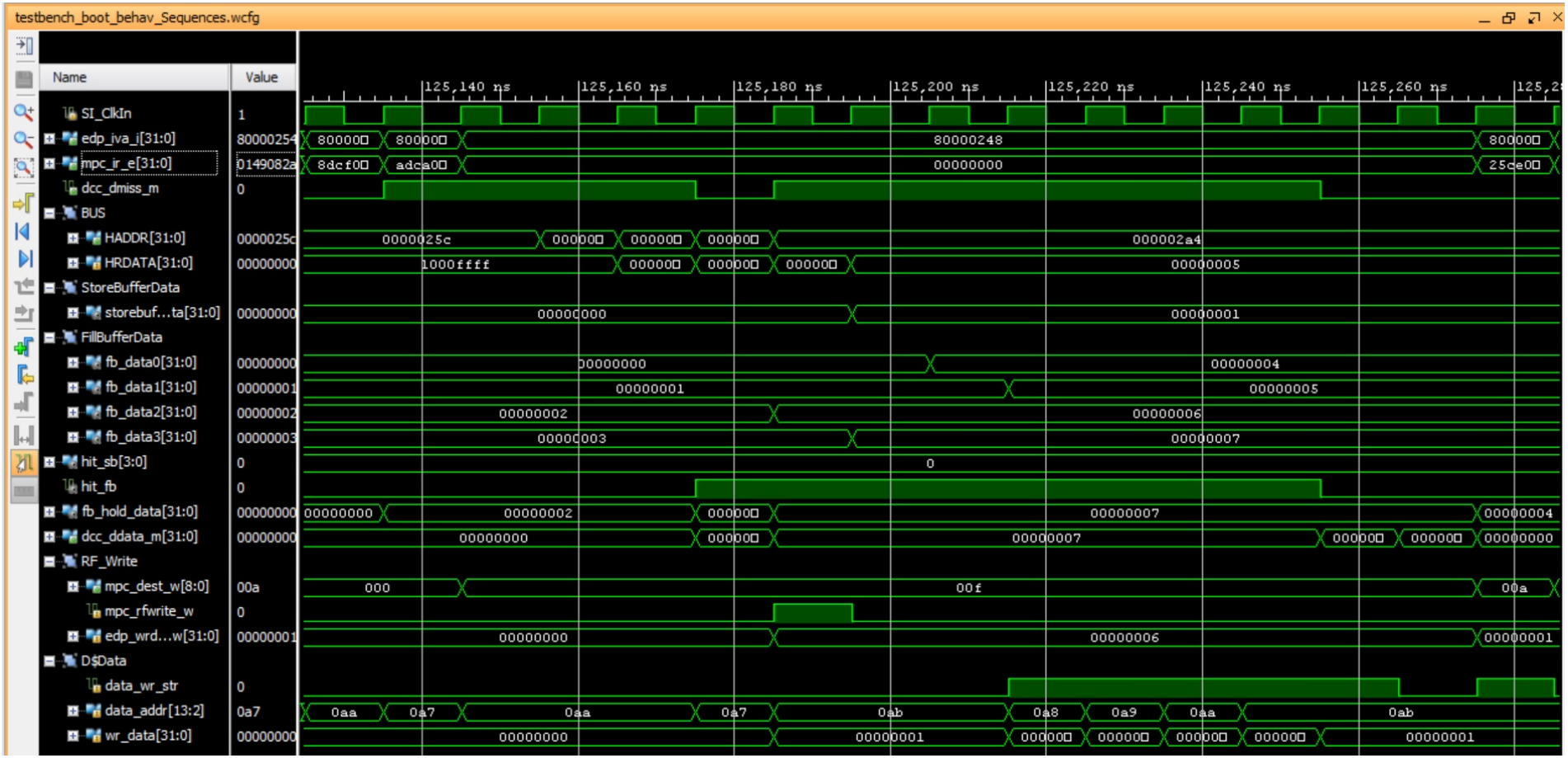
The first `lw` pays the miss penalty (4 cycles), but the second `lw` receives the requested word directly from the Fill Buffer, thus it pays a smaller or even non-existent penalty. The results of the simulation are provided in the next figure.

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


testbench_boot_behav_Sequences.wcfg
Name
Value
125,060 ns
125,070 ns
125,080 ns
125,090 ns
125,100 ns
125,110 ns
125,120 ns
125,130 ns
SI_ClkIn
edp_iva_i[31:0]
mpc_ir_e[31:0]
dcc_dmiss_m
BUS
HADDR[31:0]
HRDATA[31:0]
StoreBufferData
storebuff_data[31:0]
FillBufferData
fb_data0[31:0]
fb_data1[31:0]
fb_data2[31:0]
fb_data3[31:0]
hit_sb[3:0]
hit_fb
fb_hold_data[31:0]
dcc_ddata_m[31:0]
RF_Write
mpc_dest_w[8:0]
mpc_rfwrite_w
edp_wrdata_w[31:0]
D$Data
data_wr_str
data_addr[13:2]
wr_data[31:0]

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$.


testbench_boot_behav_Sequences.wcfg
Name
Value
125,140 ns
125,160 ns
125,180 ns
125,200 ns
125,220 ns
125,240 ns
125,260 ns
SI_ClkIn
edp_iva_i[31:0]
mpc_ir_e[31:0]
dcc_dmiss_m
BUS
HADDR[31:0]
HRDATA[31:0]
StoreBufferData
storebuf...ta[31:0]
FillBufferData
fb_data0[31:0]
fb_data1[31:0]
fb_data2[31:0]
fb_data3[31:0]
hit_sb[3:0]
hit_fb
fb_hold_data[31:0]
dcc_ddata_m[31:0]
RF_Write
mpc_dest_w[8:0]
mpc_rfwrite_w
edp_wrd...w[31:0]
D$Data
data_wr_str
data_addr[13:2]
wr_data[31:0]