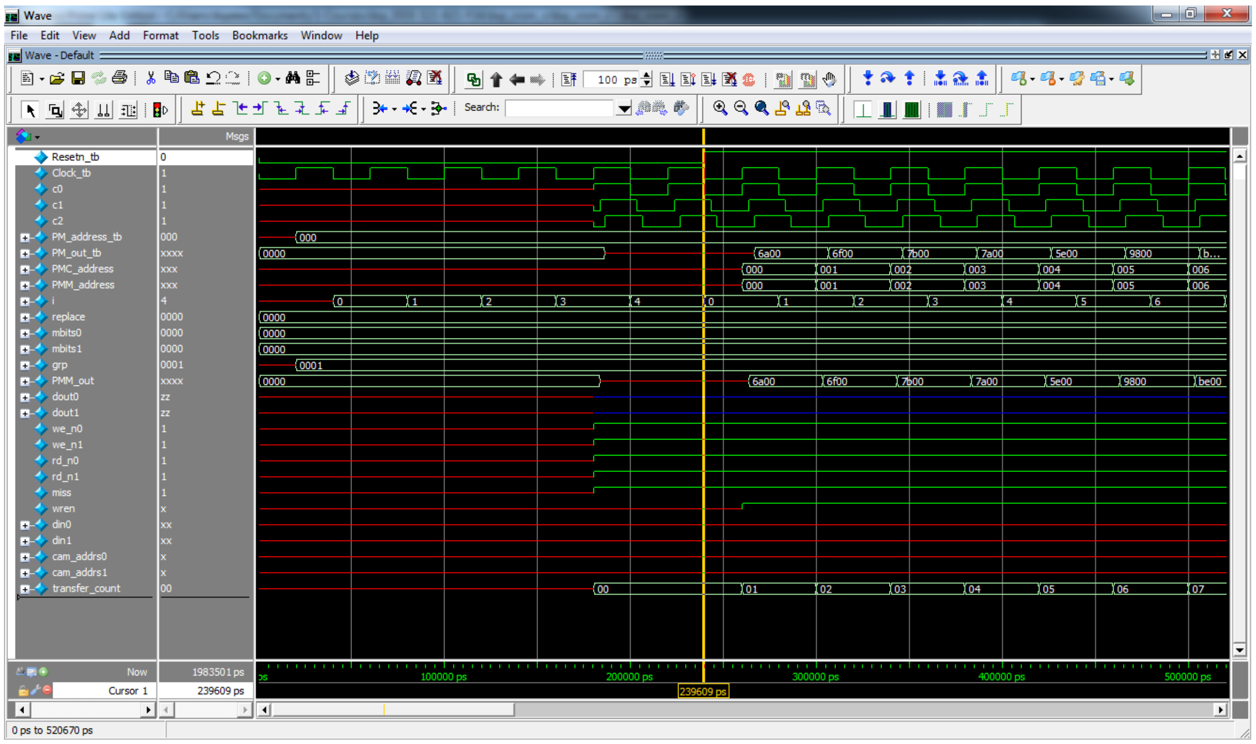
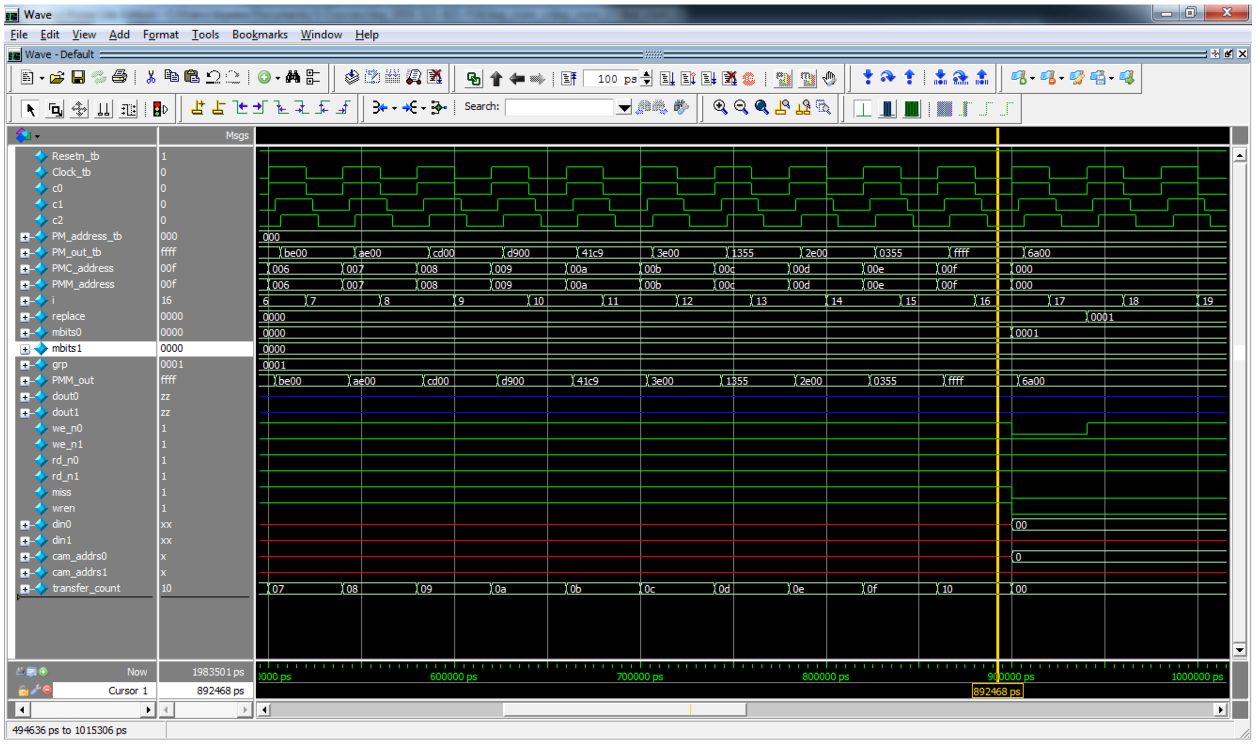
EEEE-521-621-Lab9

1) From mycourses download on your desktop this pdf file.
2) The goal of this lab is to convert your "monolithic" blocks of memory (PM and DM or MM) into a hierarchical memory blocks as follows:
   a. Harvard:
      i. PM ➔ Instruction Cache and PM
      ii. DM ➔ Data Cache and DM.
      iii. The two hierarchical memory blocks are identical.
   b. Von Neumann: MM ➔ Unified Cache and MM.
3) If you don't have a working pipelined CPU, see grading scheme at the end of this document.
4) In the case of a cache HIT, the access time is one cycle, exactly as it was before with the monolithic memory block. However, in the case of a miss, either read or write, the block will have to be transferred into the cache. In principle, the speed penalty is as follows:
   a. First cycle to determine that there is a miss.
   b. Next 8(16) cycles write back the block chosen for replacement into the memory. This is only necessary if the Dirty Bit is set.
   c. Next 8 (16) cycles read the required block into the cache.
   d. Last cycle to repeat the access to the cache and obviously HIT and complete it.
5) As you will see in the attached hierarchical memory template, you may need to use two or three different clocks. These have the same frequency, but different phases: 0, 30, and 60 degrees. As you will see in this template, this clocks are necessary to sequence the data transfers between the cache and the memory, while allowing sufficient time for the control to complete its decision making process.
6) You can generate these clocks by going to Tools > Mega Wizard > Create a New Megafunction Variation > I/O > ALTPLL. The frequencies of the input and output clocks is 50 MHz.
7) The hierarchical memory is built using a combination of structural components (memory subsystem data path portion) and procedural statement (memory subsystem control unit). While the code is not long, it is still difficult to develop and debug due to necessary control. The goal is to make this hierarchical memory component a one-to-one replacement for the monolithic memory you have used so far. However, you are allowed to enhance the interface with additional signals, such as Reset, Access_Complete, etc., as you see fit and necessary.
8) Therefore, the only change you'll need to make to your CPU (non-pipelined or pipelined) is the ability to STALL for the number of cycles necessary to handle a miss.
9) The specifications of your hierarchical memory design should be inferred from the previous lab assignment.
10) To demonstrate that your design is working, copy/paste the current content of your *.mif file at the next 15 locations in sequence. As you'll see further below, your grade is based on demonstrating a working: Read Hit, Write Hit, Read Miss, and Write Miss. Modify the code in any way you need to be able to trigger all these events. It obviously depends on your cache specifications.

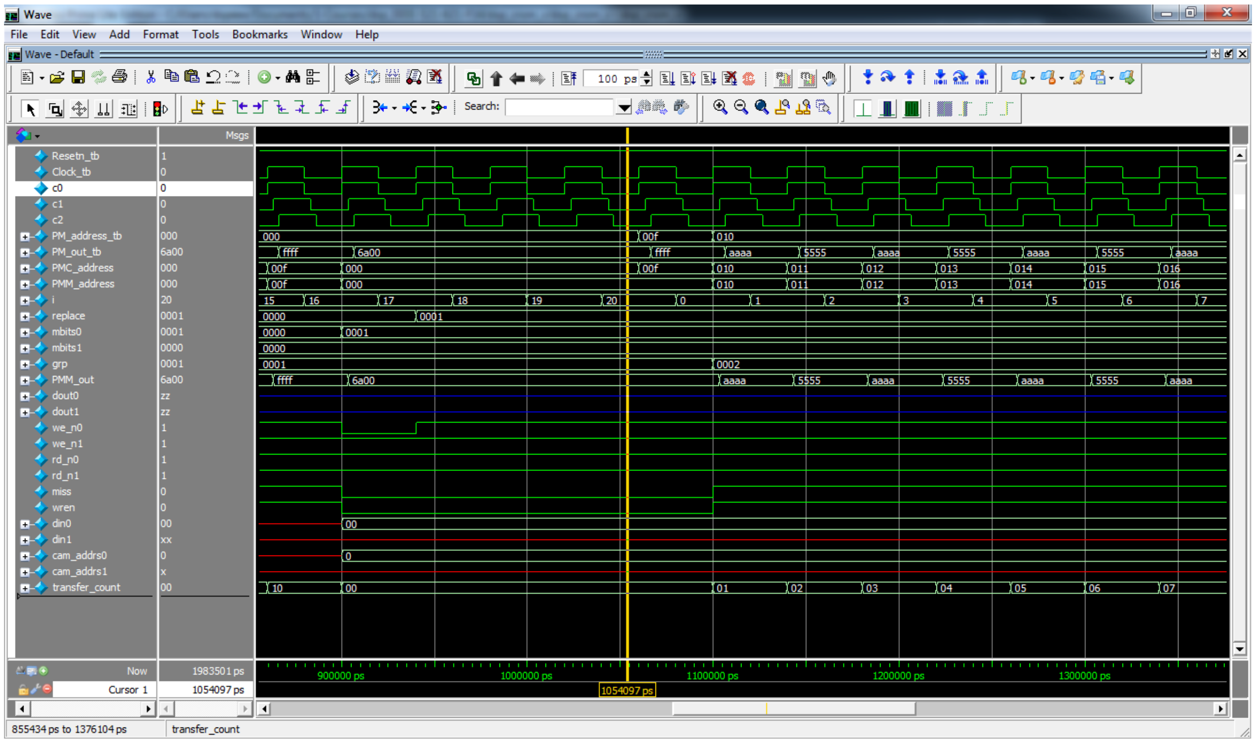
11) I am providing the Verilog version of the hierarchical memory for a ROM based PM. You'll have to:
    a. Convert this to your specifications.
    b. Harvard: create a similar one and add the write feature.
    c. Von Neumann: add the write feature.
12) Your current memory doesn't need changes.
13) However, your cache will need to be created using the wizard again.
14) Once the tag memory determines a match, it generates an address that points to the respective block location in the cache. This address value concatenated with the word address field points finally to the right word in the cache.
15) The rest of the hierarchical memory block organization can be inferred from the cache presentation used in the lecture and lab.
16) If time permits, show your simulation to your TA. Note: only the TA will assign a grade.
17) Archive your project.
18) Write your report and upload it along with your archived project(s) in the dropbox on mycourses, as described in the lab policy.
19) This concludes this week's lab.
20) **Grading**:
    a. 12 points for demonstrating a correct Read Hit.
    b. 12 points for demonstrating a correct Write Hit.
    c. 13 points for demonstrating a correct Read Miss.
    d. 13 points for demonstrating a correct Write Miss.
    e. -10 points penalty if demonstrated with the non-pipelined FML RISC processor.
21) **Operation**:
22) The testbench holds the system in reset for approximately 6 clock periods.
23) After reset, it transfers the 16 words of the first block into the cache.


```
Resetn_tb       0
Clock_tb        1
c0              1
c1              1
c2              1
PM_address_tb   000
PM_out_tb       xxxx
PMC_address     xxx
PMM_address     xxx
i               4
replace         0000
mbits0          0000
mbits1          0000
grp             0001
PMM_out         xxxx
dout0           zz
dout1           zz
we_n0           1
we_n1           1
rd_n0           1
rd_n1           1
miss            1
wren            x
din0            xx
din1            xx
cam_addrs0      x
cam_addrs1      x
transfer_count  00

Now       1983501 ps
Cursor 1  239609 ps
0 ps to 520670 ps
```

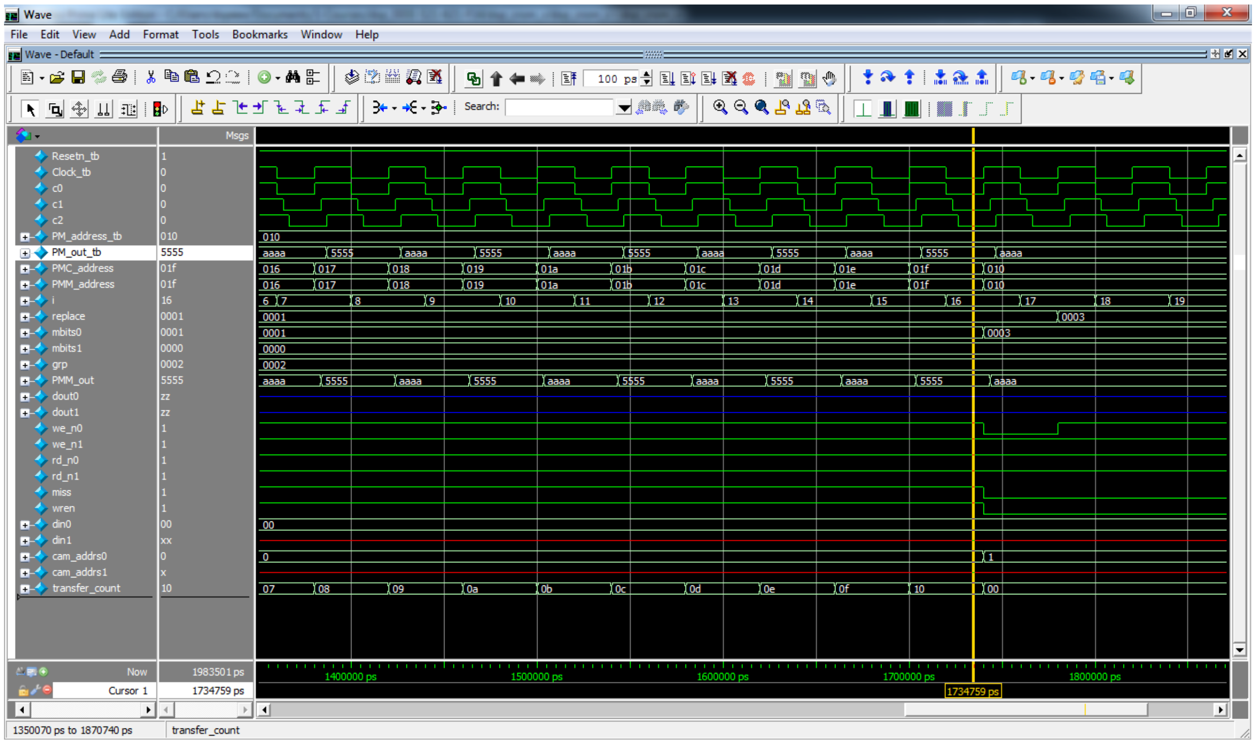
24) At 900 ns it is ready to output to the CPU the value of the first word, which in this case is “6a00”.


```
Resetn_tb       1
Clock_tb        0
c0              0
c1              0
c2              0
PM_address_tb   000
PM_out_tb       ffff
PMC_address     00f
PMM_address     00f
i               16
replace         0000
mbits0          0000
mbits1          0000
grp             0001
PMM_out         ffff
dout0           zz
dout1           zz
we_n0           1
we_n1           1
rd_n0           1
rd_n1           1
miss            1
wren            1
din0            xx
din1            xx
cam_addrs0      x
cam_addrs1      x
transfer_count  10

Now       1983501 ps
Cursor 1  892468 ps
494636 ps to 1015306 ps
```

25) Four cycles later the testbench, i.e. the CPU, requests a word from address 0x00f, which is still a hit because it is the last word of the first block that was brought into the cache.

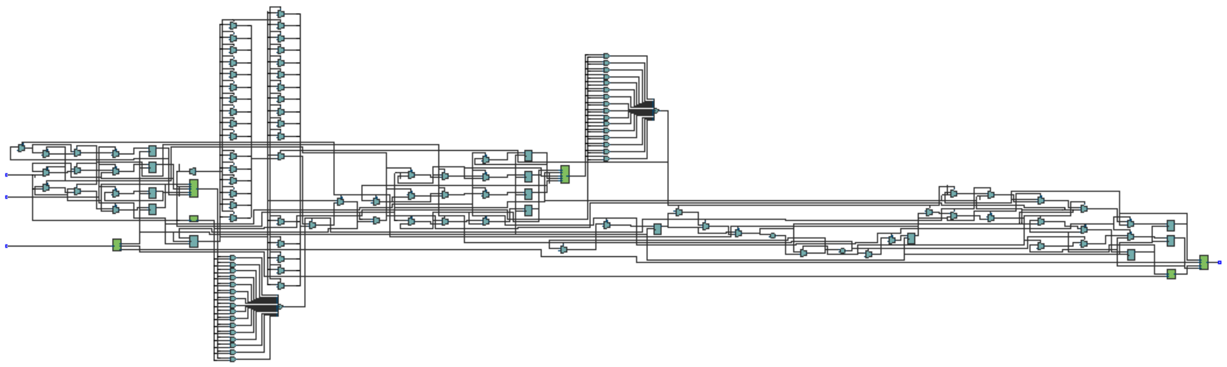


26) However, one cycle later, the testbench, i.e. the CPU, requests a word from address 0x010, which is the first word of the second block that is not yet in the cache. This is obviously a miss, and thus the system transfers the second block into the cache during the next 16 cycles.

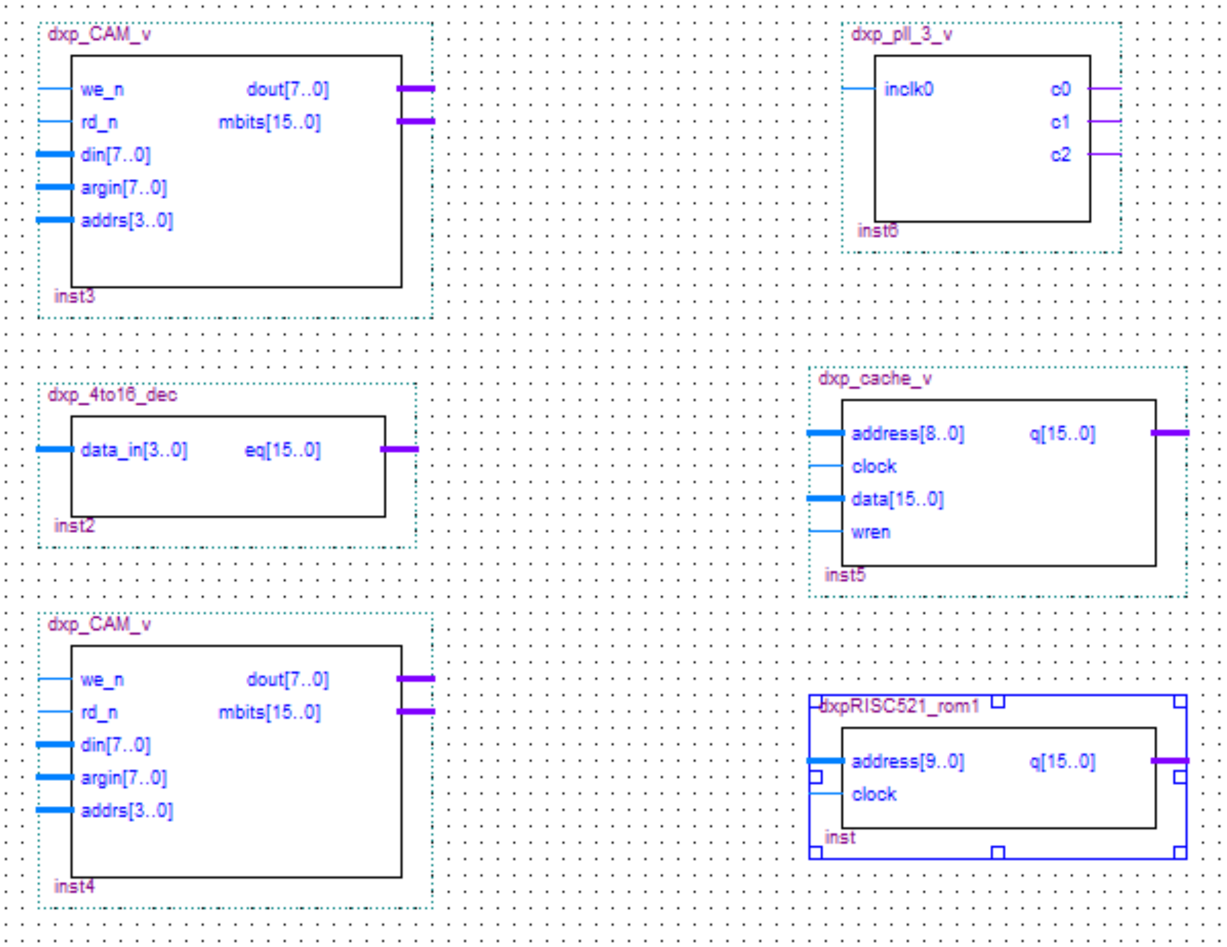


27) Finally, the system outputs the value “aaaa” at address 0x010 as a hit.

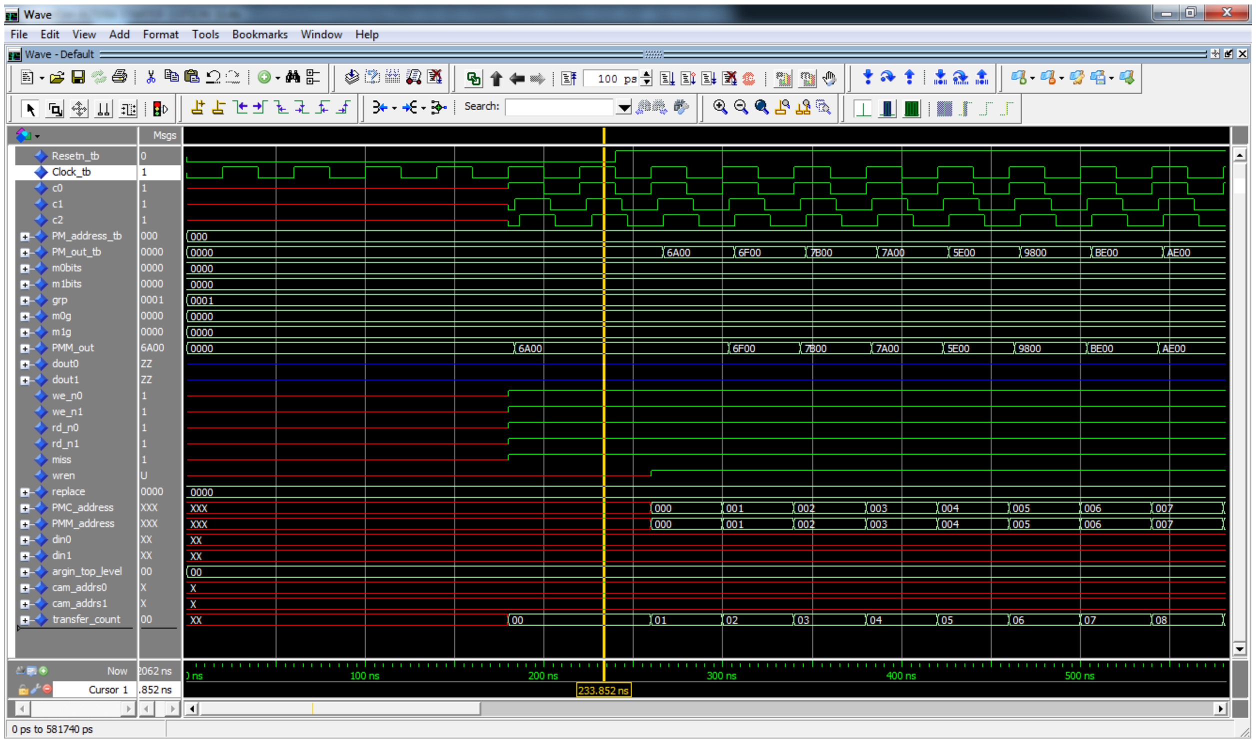
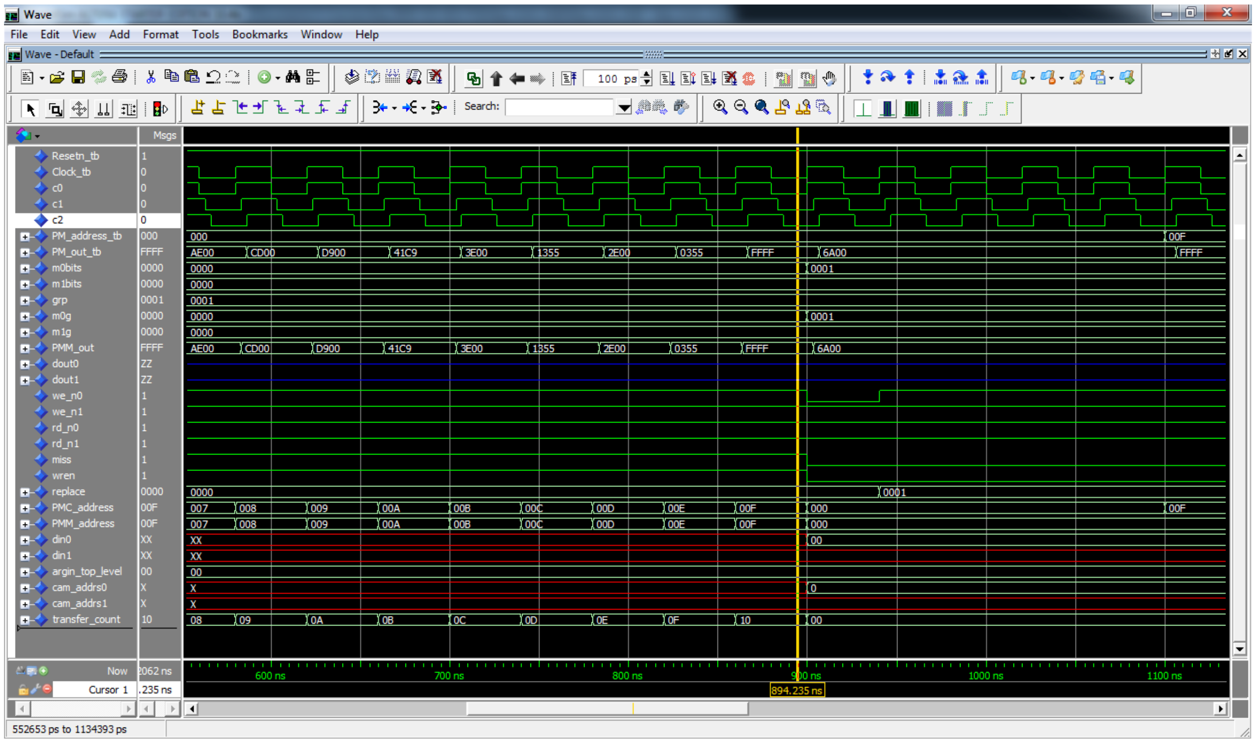
28) The synthesized circuit:

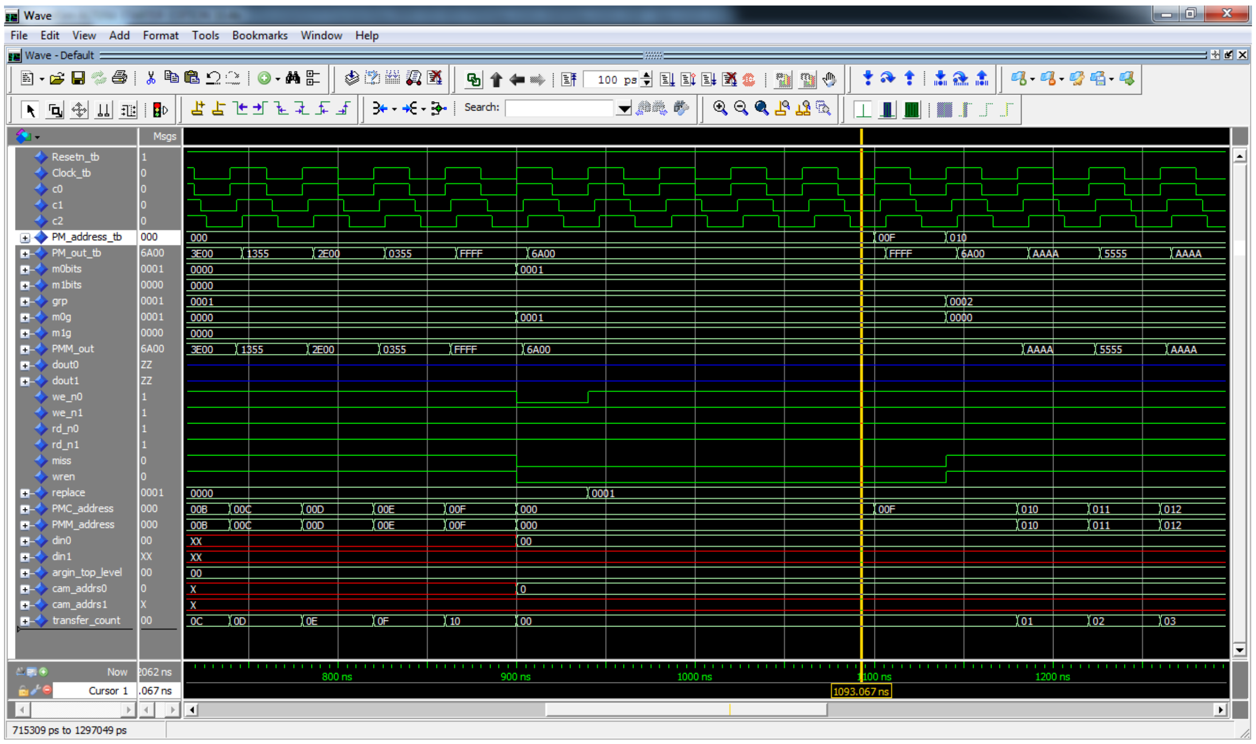


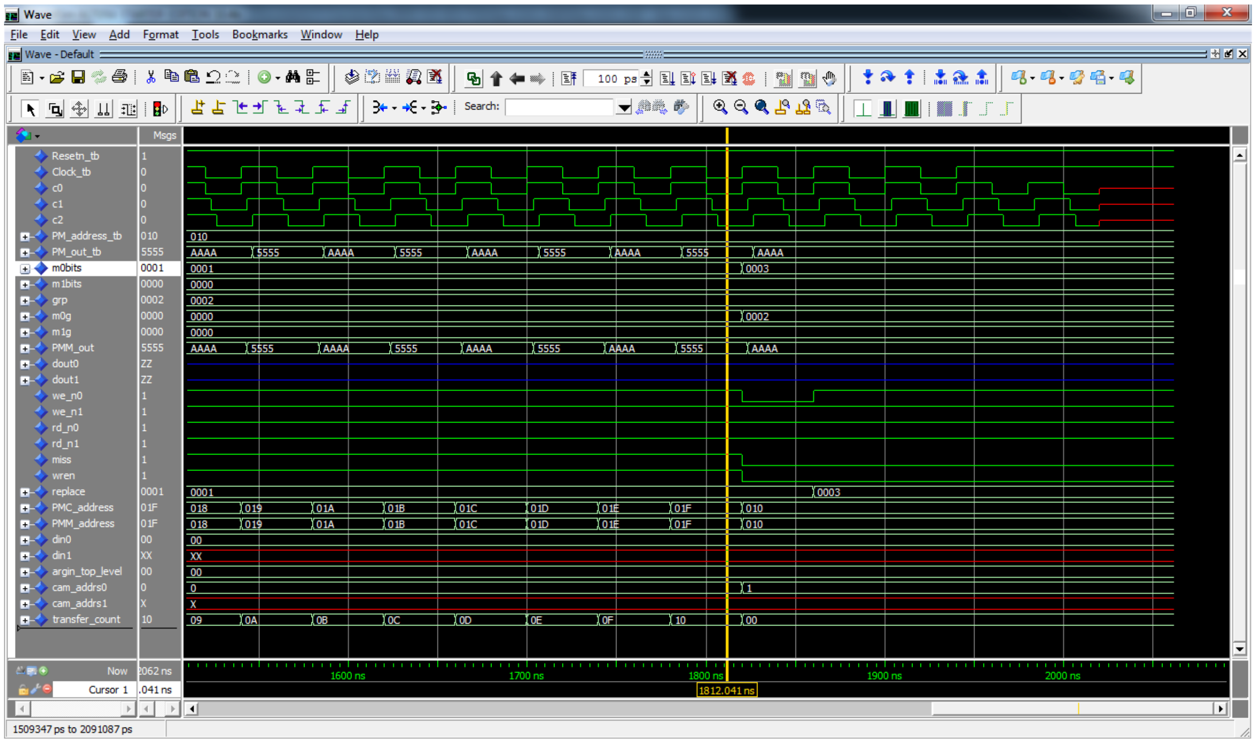
29) And the individual functional blocks:

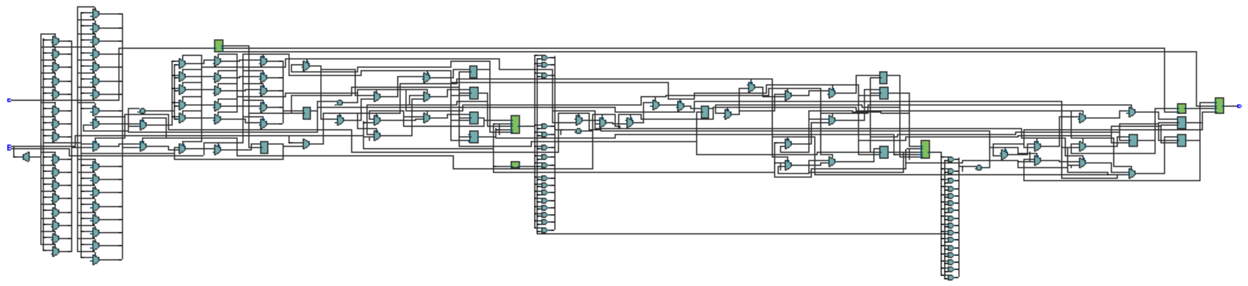


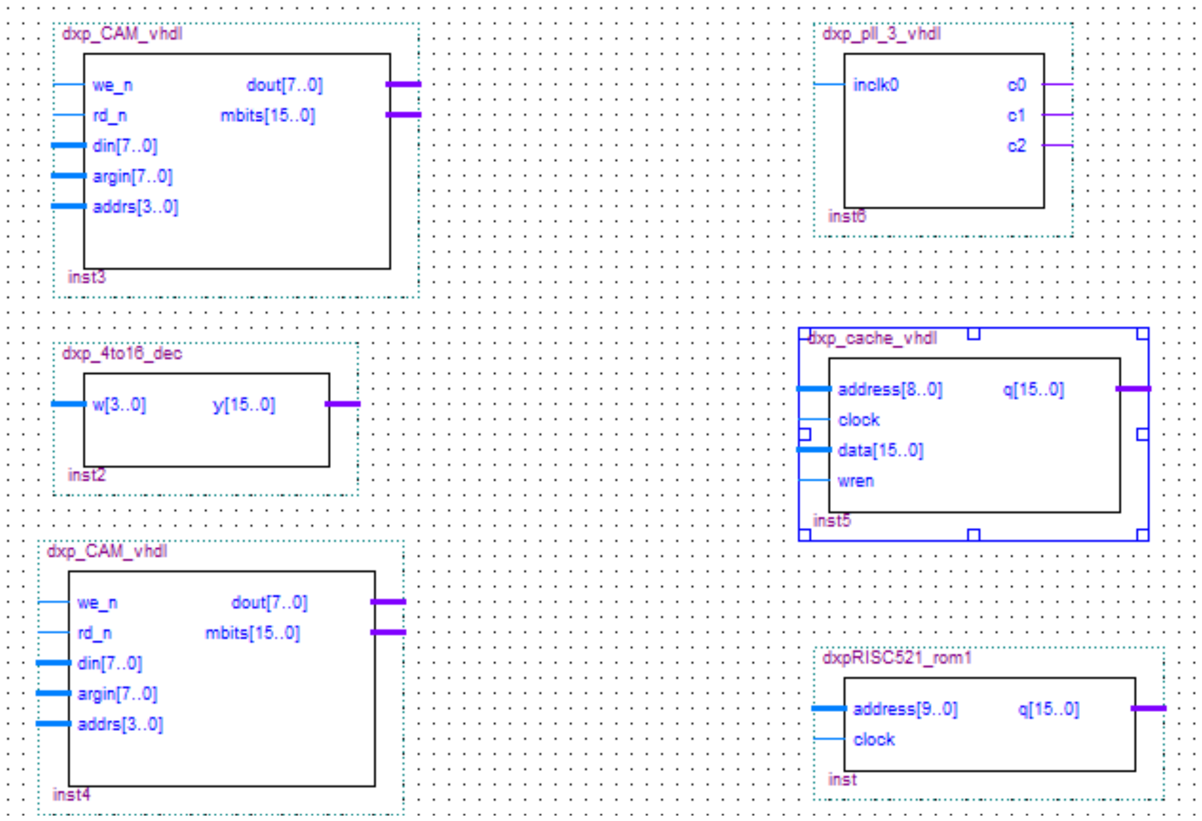
30) Below are the same relevant screen captures from the VHDL implementation:


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
Resetn_tb 0
Clock_tb 1
c0 1
c1 1
c2 1
PM_address_tb 000
PM_out_tb 0000
m0bits 0000
m1bits 0000
grp 0001
m0g 0000
m1g 0000
PMM_out 6A00
dout0 ZZ
dout1 ZZ
we_n0 1
we_n1 1
rd_n0 1
rd_n1 1
miss 1
wren U
replace 0000
PMC_address XXX
PMM_address XXX
din0 XX
din1 XX
argin_top_level 00
cam_addrs0 X
cam_addrs1 X
transfer_count 00
6A00 6F00 7B00 7A00 5E00 9800 BE00 AE00
000 001 002 003 004 005 006 007
00 01 02 03 04 05 06 07 08
Now 2062 ns
Cursor 1 .852 ns
0 ns 100 ns 200 ns 300 ns 400 ns 500 ns
233.852 ns
0 ps to 581740 ps

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
Resetn_tb 1
Clock_tb 0
c0 0
c1 0
c2 0
PM_address_tb 000
PM_out_tb FFFF
m0bits 0000
m1bits 0000
grp 0001
m0g 0000
m1g 0000
PMM_out FFFF
dout0 ZZ
dout1 ZZ
we_n0 1
we_n1 1
rd_n0 1
rd_n1 1
miss 1
wren 1
replace 0000
PMC_address 00F
PMM_address 00F
din0 XX
din1 XX
argin_top_level 00
cam_addrs0 X
cam_addrs1 X
transfer_count 10
AE00 CD00 D900 41C9 3E00 1355 2E00 0355 FFFF 6A00
0001
007 008 009 00A 00B 00C 00D 00E 00F 000
08 09 0A 0B 0C 0D 0E 0F 10 00
Now 2062 ns
Cursor 1 .235 ns
600 ns 700 ns 800 ns 900 ns 1000 ns 1100 ns
894.235 ns
552653 ps to 1134393 ps


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
Resetn_tb 1
Clock_tb 0
c0 0
c1 0
c2 0
PM_address_tb 000
PM_out_tb 6A00
m0bits 0001
m1bits 0000
grp 0001
m0g 0001
m1g 0000
PMM_out 6A00
dout0 ZZ
dout1 ZZ
we_n0 1
we_n1 1
rd_n0 1
rd_n1 1
miss 0
wren 0
replace 0001
PMC_address 000
PMM_address 000
din0 00
din1 XX
argin_top_level 00
cam_addrs0 0
cam_addrs1 X
transfer_count 00
Now 2062 ns
Cursor 1 .067 ns
800 ns
900 ns
1000 ns
1100 ns
1200 ns
1093.067 ns
715309 ps to 1297049 ps


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
Resetn_tb 1
Clock_tb 0
c0 0
c1 0
c2 0
PM_address_tb 010
PM_out_tb 5555
m0bits 0001
m1bits 0000
grp 0002
m0g 0000
m1g 0000
PMM_out 5555
dout0 ZZ
dout1 ZZ
we_n0 1
we_n1 1
rd_n0 1
rd_n1 1
miss 1
wren 1
replace 0001
PMC_address 01F
PMM_address 01F
din0 00
din1 XX
argin_top_level 00
cam_addrs0 0
cam_addrs1 X
transfer_count 10
Now 2062 ns
Cursor 1 .041 ns
1600 ns
1700 ns
1800 ns
1900 ns
2000 ns
1812.041 ns
1509347 ps to 2091087 ps




dxp_CAM_vhdl
we_n
rd_n
din[7..0]
argin[7..0]
addrs[3..0]
dout[7..0]
mbits[15..0]
inst3
dxp_pll_3_vhdl
inclk0
c0
c1
c2
inst6
dxp_4to16_dec
w[3..0]
y[15..0]
inst2
dxp_cache_vhdl
address[8..0]
clock
data[15..0]
wren
q[15..0]
inst5
dxp_CAM_vhdl
we_n
rd_n
din[7..0]
argin[7..0]
addrs[3..0]
dout[7..0]
mbits[15..0]
inst4
dxpRISC521_rom1
address[9..0]
clock
q[15..0]
inst

31) The END ;-)