Case C.

1) Remove the performance counter unit from your nios-system and re-add the interval timer to your system. Make sure to setup the connections as you did in Project 5 (including the interrupt priority).
2) Add a second Nios processor to your system so you should have two processors in your system.
    a. Each processor must be self-identified to behave accordingly (Open nios processor -> Advance Features -> CPUID control value)
        i. Processor 0 → 0x00000000
        ii. Processor 1 → 0x00000001
        iii. Reset and exception vectors of Processor 1 (0x00000000, 0x00000020 respectively)
        iv. Reset and exception vectors of Processor 2 (0x08000000, 0x08000020 respectively)
3) Add a second JTAG UART to the system which will be connected to the second processor.
4) Add a second timer component which will be connected to the second processor (with same connection setup as the first one, including the interrupt settings). Both timers must be setup on 100 μs. Set the priority of the timer components as 0 and the priority of the JTAG UART as 16 for each processor.
5) Connect the two processors to the same SDRAM component by connecting the instruction and data master of the processors to the SDRAM_slave.
6) Add one Mutex Component and one on-chip memory to the system (32-bit width and 4096 memory size). Connect both to the data master of the two processors.
7) No changes need to be done in Test.v.
8) Re-generate the system, recompile Quartus project, and configure the board.
9) Re-generate the system, compile, and synthesize your project. Read the summary report from Quartus and fill out the below table. Save the Table in the report file.

Table C: Custom_Floating_point_Timer_Multiprocessor

Logical Elements	Registers	Total Pins	Memory Bits

Application Project

1. In addition to the existing project, create another application project (complete project includes app folder and bsp folder) for the second processor.
2. The two applications should have the identical code, except for-loop (i) and the processor id that should match the processor that is targeted in each application.
3. The for-loop represents the image row which will be divided between two processors.
4. Open the two application projects and right-click on their BSP directories, go to Nios II and select Nios II editor. Set the following features, then regenerate the BSP.

Processor	Sys_clk_timer	Timestamp_timer	stdin	stdout	stderr
0	Timer_0	none	jtag	jtag	jtag
1	Timer_1	none	jtag	jtag	jtag

4. In each application directory, open the main (or hello_world.c) program. Don't forget to add the following headers:

```
#include <sys/alt_alarm.h>
#include <altera_avalon_mutex.h>
#include <io.h>
```

6. In the frame_function () of processor0-application, change only the row for-loop into

```
for(int i=0; i<row; i+=2)
```

7. In the frame_function() of processor1-application, change only the row for-loop into

```
for(int i=1; i<row; i+=2)
```

8. Add the barrier_function to your code that is provided in L9. Then, call the barrier function in the main program of each application before the frame_function ().

9. Measure the time (number of ticks) required to execute the average frame time for one of the processor as you did in Project 5.